Computer Validated Proofs of a Toolset for Adaptable Arithmetic

Marc Daumas, Claire Moreau-Finot, Laurent Théry

INSTITUT NATIONAL DE RECHERCHE EN INFORMATIQUE ET EN AUTOMATIQUE

Computer validated proofs of a toolset for adaptable arithmetic

Marc Daumas, CNRS
Claire Moreau-Finot, LIP
Laurent Théry, INRIA

No 4095

Janvier 2001

THÈME 2

ISSN 0249-6399 ISRN INRIA/RR--4095--FR+ENG

RHÔNE-ALPES

Computer validated proofs of a toolset for adaptable arithmetic

Marc Daumas, CNRS
Claire Moreau-Finot, LIP
Laurent Théry, INRIA

Thème 2 — Génie logiciel
et calcul symbolique
Projet Arénaire

Rapport de recherche n ° 4095 — Janvier 2001 — 29 pages

Abstract: Most existing implementations of multiple precision arithmetic demand that the user sets the precision *a priori*. A solution is to use the largest precision necessary to reach target accuracy. Some libraries are said adaptable in the sense that they dynamically change the precision of each intermediate operation individually to deliver the target accuracy according to the actual inputs. We present in this text a new adaptable numeric core inspired both from floating point expansions and from on-line arithmetic.

The numeric core is cut down to five tools. The first tool that contains many arithmetic operations is proved to be correct. The proofs have been formally checked by the Coq assistant. Developing the proofs, we have formally proved many result published in the literature and we have extended a few of them. This work may let users (i) develop application specific adaptable libraries based on the toolset and / or (ii) write new formal proofs based on the set of validated facts.

Key-words: Multiple precision, expansion, on-line, formal proof, Coq.

(Résumé : tsvp)

This text is also available as a research report of the Laboratoire de l'Informatique du Parallélisme `http://www.ens-lyon.fr/LIP`.

Unité de recherche INRIA Rhône-Alpes
655, avenue de l'Europe, 38330 MONTBONNOT ST MARTIN (France)
Téléphone : 04 76 61 52 00 - International: +33 4 76 61 52 00
Télécopie : 04 76 61 52 52 - International: +33 4 76 61 52 52

Boîte à outils pour l'arithmétique numérique validée par un preuve sur ordinateur

Résumé : La plupart des codes à précision multiple existants demandent que l'utilisateur spécifie la précision des calculs *a priori*. Une solution est d'utiliser pour les calculs intermédiaires la précision la plus élevée pour atteindre la précision demandée pour le résultat. On dit que certaines bibliothèques sont adaptatives dans le sens qu'elles changent dynamiquement la précision de chacun des calculs intermédiaires en fonction des valeurs effectives des entrées pour atteindre la précision demandée pour le résultat. Nous présentons dans ce texte un nouveau noyau numérique adaptatif inspiré à la fois des expansions de nombres flottants et de l'arithmétique en-ligne.

Le noyau numérique se décompose en cinq outils. Le premier outil qui contient de nombreuses opérations arithmétiques est prouvé. Les preuves ont été validées formellement par l'assistant Coq. En développant les preuves, nous avons redéveloppé formellement de nombreux résultats déjà publiés dans la littérature et nous en avons étendu certains. Ce travail permettra aux utilisateurs de (i) développer des bibliothèques adaptatives spécifiques à une application basées sur la boîte à outils et / ou (ii) écrire de nouvelles preuves formelles basées sur notre socle de faits validés.

Mots-clé : Précision multiple, expansion, en-ligne, preuve formelle, Coq.

Introduction

Motivation

As an undergraduate student, we have all learnt to compute a quantity with a few significant digits first as 5 miles is about 8 kilometers. If the question is "Are 4.995 miles more than 8 kilometers?", we are close to a threshold. To answer the question, we would compute more (less significant) digits to find that 4.995 miles are a little over 38 meters longer than 8 kilometers. We might first do a multiplication of 5 by 1.6 and then a multiplication of 4.995 by 1.609344 with a higher working precision than the first time. However, we could save that 5 times 1.6 is exactly 8 and afterward just multiply 5 by 9 to obtain 45 with a small additional value that is less than 5 meters. That answers the question since 0.005 miles is less than 10 meters.

Numerical analysts have long designed algorithms that react to the accuracy aimed by the user. Specific algorithms reduce the error on the final result by iterative refinement in solving algebraic equations, linear systems, ODEs, PDEs and so on [27]. Such methods have been so successful that the final solution can be very accurate even if the first answer before any refinement is not close to it. All the above-mentioned solutions assume that the intermediate arithmetic operations are performed up to a fixed level of precision. This level, usually the one given by machine double precision arithmetic, is used for all the operations and is sufficient to store the result to the accuracy aimed by the user. Recent work presented in [27, 51, 5] proposes new enhanced routines if two levels of precision are available on the machine. All these algorithms use the machine arithmetic as an error prone tool that must be circumvented by mathematical analysis.

Prior art in adaptability

Coherency is key to the field of computational geometry where a single tiny error may change a convex hull to a non-convex, possibly non-planar, graph. Numerical routines are used to answer such predicates as "Is a point in a circle defined by 3 other points?". A Boolean answer cannot be approximated, it is either right or wrong, and a wrong answer might lead a program to an abstract state that cannot occur in Euclidian geometry. Recent works have proposed clever implementations of multiple precision arithmetic [22, 31] to answer correctly such questions. They reinforce the existing set of general purpose multiple precision libraries [10, 48, 3, 4, 26]. Some marginal work does implement iterative refinements of the geometric predicates [2] or change the algorithm to work correctly with limited predicates [9].

Adaptability may be defined at the *application level*, as this is the case for all the above-mentioned solutions, or at the *arithmetic level*, as we will see in the remaining of this text. The application is only slightly modified to accommodate the higher working precision and the benefit in accuracy on the final result is mostly obtained by the higher accuracy of the intermediate results. It is started with a very limited precision but every single evaluation continues with the library increasing the intermediate working precision until the final result is known to a satisfying precision. The two domains interacted as Pichat presented an adaptable algorithm to compute an accumulation of numbers [45] by adapting the one-step enhancement already presented in [42, 41, 30].

Few authors already presented a general purpose adaptable arithmetic library [7, 38, 39, 6, 40, 11], but some others stopped with one or two adaptable attempts followed by an exact evaluation [22, 50]. These last solutions are appropriate in many cases because the first few attempts are fast and they succeed by far in the most frequent case for many applications [19]. These authors where forced to stop because they used a data structure that is not appropriate for adaptability. They only squeezed in one or two rounds of adaptability with great efforts.

We present in this work a data structure inspired by on-line arithmetic where adaptability is natural. On line arithmetic was targeted in the past to hardware design with a small radix (typically 2 or 4) [53, 12, 34, 17, 16] but it has proved to be suitable for software applications [37, 6].

Expansions were introduced by Priest [46] based on some earlier operations [32, 33]. Actually, Møller and Dekker were the first in the past to propose such techniques [42, 41, 18]. More recently, IEEE standard commercial floating point units lead to faster algorithms more tightly connected to the machines. Shewchuk was the first one to present a working library available on the net with an actual application to computational geometry [49, 50]. Assuming that the floating point unit is IEEE compatible gives us a powerful set of axiomatic properties on floating point operations. We have proposed in the past some algorithms and their

implementation for multiplication and division on expansions [13, 14, 15]. Developments are still undergone as new research teams get interested by this approach [47].

Our contribution forces us to define a new kind of expansions we call pseudo-expansions. This definition yields a more redundant notation [35, 44]. Although the proof of [36] does not apply here, it is sensible that more redundancy in the notation eases the implementation of most significant digit first (MSDF) arithmetic [1, 43, 23].

Contributions

The first contribution of this work is a set of primitives available soon on the internet through our home web site[1] and through the NETLIB repository[2]. A toy example shows that high level programming with object interface is too slow for the kernel of numerical softwares [24]. We ran the same FLOPS routines three times [20, 21]. The first run was performed with object automatic allocation and destruction. The second run used inline function calls and no object allocation. The final code used explicit code inlining. Object programming caused a decrease of 133% of the performance and clean code inlining caused a decrease of 56%. Our contributed building blocks are pieces of code assembled by the designer according to its needs. They present a straight interface to get a good trade off between simplicity and speed.

The second contribution is a set of validated proofs. To mechanize our proofs, we have been using the Coq proof assistant [28]. Systems like Coq allow the user to define new objects and to derive consequences of these definitions formally. The language of Coq is based on a higher-order logic. With such an expressive logic, it is possible to state properties in their most general form. For example, universal quantification has been used to state properties that are true for an arbitrary format or an arbitrary rounding mode. Proofs are built interactively using high-level tactics. At the end of each proof, Coq records a proof object that contains all the details of the derivation and ensures that the theorem is valid.

Our formal development is freely available on the internet[3] and it will soon be available as a Coq contribution[4]. A clickage map of the hierarchy makes it possible to browse into the different components. At the moment, it is 10000 line long and contains 77 Definitions and 538 Theorems. No proof is presented in this manuscript. All the Definitions and Theorems of this development are presented in annexes A through W. For each mentioned theorem we give its name as it appears in the formal development and the file where it is proved.

We first present definitions and validated properties for the floating point numbers, the expansions and the pseudo-expansions. Section 2 presents the addition, multiplication and division toolset. This presentation ends with concluding remarks in the last Section.

1 Definitions and validated properties

1.1 Floating point numbers

An IEEE normalized double precision floating point number is built from 3 binary fields: the sign (1 bit), the fraction (52 bits) and the biased exponent (11 bits). Its interpretation as a rational number is given below.

$$x = (-1)^{\text{sign}} \times \text{mantissa} \times 2^{\text{exponent}} \quad \text{with} \quad \text{mantissa} = 1.\text{fraction} \quad \text{and} \quad \text{exponent} = \text{biased exponent} - \text{bias}$$

We will use in this text the more general notation $x = n \times \beta^e$ to define a floating point number of radix β with two integers, the significand n and the ulp e. This relation gives a signification to any pair $u = (n, e) \in \mathbb{Z}^2$ and we denote $\mathbb{F}$ the set $\mathbb{Z}^2$ with the canonical mapping into the rational numbers. Two pairs u and v are equivalent if they have the same signification, we write $u \equiv v$. Given a pair $\mathcal{B} = (n_{max}, e_{min}) \in \mathbb{N}_*^2$, we say that a pair (n, e) is bounded if and only if it satisfies

$$|n| \leq n_{max} \qquad \text{and} \qquad -e_{min} \leq e. \tag{1}$$

[1] `http://www.ens-lyon.fr/LIP/Arenaire/`.
[2] `http://www.netlib.org/` among many access protocols.
[3] `http://www-sop.inria.fr/lemme/AOC/coq/`.
[4] `http://coq.inria.fr/`.

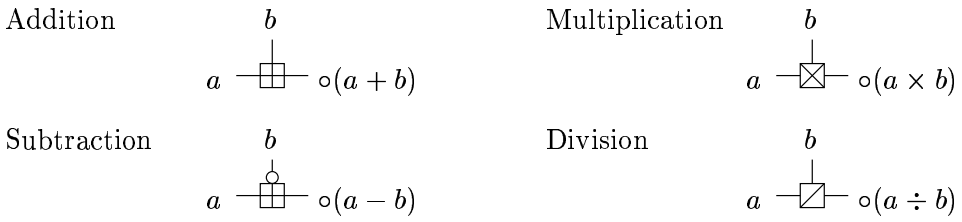


Figure 1: Standard floating point operations rounded to the nearest (even) value.

Equation (1) defines the set $\mathbb{F}_{\mathcal{B}}$ of the pairs bounded by $\mathcal{B}$. The bound for an IEEE standard inspired representation $\mathcal{R}$ with $m_{\mathcal{R}}$ digits of mantissa and $e_{\mathcal{R}}$ digits of exponent is $(\beta^{m_{\mathcal{R}}} - 1, \lceil \beta^{e_{\mathcal{R}}}/2 \rceil + m_{\mathcal{R}} - 3)$. The usual fraction interpretation is easily obtained dividing the integer significand by $\beta^{m_{\mathcal{R}}}$ or $\beta^{m_{\mathcal{R}}-1}$, and changing the ulp accordingly. We will use the fraction notation for the examples although the integer notation is used for the proofs.

We do not use the overflow bound of $e_{max} = \lfloor \beta^{e_{\mathcal{R}}}/2 \rfloor - m_{\mathcal{R}}$. All the results of this text are true provided no overflow occurred in the computation. If this is not the case, infinity and *not a number* quantities will proliferate in the results.

In the machines $\beta = 2$ and $(m_{\mathcal{R}}, e_{\mathcal{R}}) = (24, 8)$ for the single precision format and $(m_{\mathcal{R}}, e_{\mathcal{R}}) = (53, 11)$ for the double precision format giving the respective bounds $(2^{24} - 1\,; 2^7 + 24 - 3) = (16\,777\,215\,; 149)$ and $(2^{53} - 1\,; 2^{10} + 53 - 3) = (9\,007\,199\,254\,740\,991\,; 1074)$.

We note $\mathbb{F}_{\mathcal{R}}$ the set of the couples bounded by such an IEEE standard inspired bound. A bounded pair is normal if $|n| \times \beta > n_{max}$. It is subnormal if it is not normal and $e = e_{min}$.

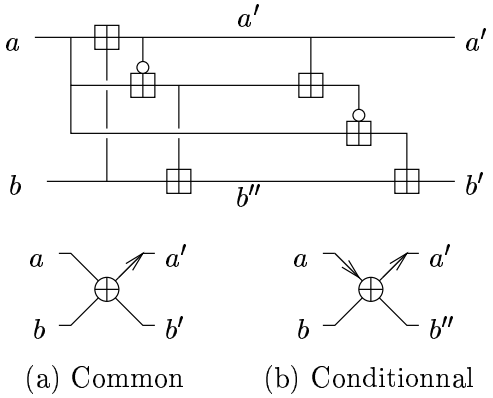
Theorem 1 (FnormalizeCanonic and FcanonicUnique in Form) *Any number $x = n \times \beta^e$ in $\mathbb{F}_{\mathcal{R}}$ is uniquely defined by either a normal or a subnormal pair (n', e') with $x \equiv (n', e')$. This pair is the direct transcription of the number in the machine and it is later referred in this text as the canonical representation.*

The IEEE standard describes four rounding modes but the rounding to the nearest floating point number is the rounding mode used by default in most computers. A rounding is a non decreasing projection on the set $\mathbb{F}_{\mathcal{B}}$. For any floating point number $x \in \mathbb{F}$ (possibly not bounded), we have defined the bounded predecessor $x^- \in \mathbb{F}_{\mathcal{B}}$ and the bounded successor $x^+ \in \mathbb{F}_{\mathcal{B}}$ as a minimum and a maximum. Any non decreasing projection $\square$ from $\mathbb{F}$ to $\mathbb{F}_{\mathcal{B}}$ uses $\square(x) \equiv x^+$ or $\square(x) \equiv x^-$. We have defined the five relations where $x \in \mathbb{F}$ and $y \in \mathbb{F}_{\mathcal{B}}$ as follows.

Rouding type	Symbol	Necessary and sufficient condition
Directed	$x\mathcal{R}_- y$	$y = x^-$
	$x\mathcal{R}_+ y$	$y = x^+$
	$x\mathcal{R}_z y$	$y = x^-$ when $x \geq 0$ and $y = x^+$ when $x \leq 0$
To the nearest	$x\mathcal{R}_\pm y$	$\forall z \in \mathbb{F}_{\mathcal{B}} \quad \lvert y - x \rvert \leq \lvert z - x \rvert$
	$x\mathcal{R}_\circ y$	$\forall z \in \mathbb{F}_{\mathcal{B}} \quad \lvert y - x \rvert < \lvert z - x \rvert$ or $x\mathcal{R}_\pm y$ and y has an even mantissa

Theorem 2 (*RoundedModeP in Fround and Closest) *For the relations rounding down, up, to zero and to the nearest (even), any floating point number has one and only one rounded value. These relations define each a total non decreasing projection from $\mathbb{F}$ to $\mathbb{F}_{\mathcal{B}}$ compatible with the interpretation of the number ($\equiv$).*

The result of any implemented operation, namely the addition, the multiplication, the division and the square root extraction, is the rounded result of the exact mathematical operation. For example, if $a \oplus b$ is the machine floating point addition and $\circ(x)$ is the rounded to nearest (even) value of x for any x, then $a \oplus b = \circ(a + b)$. Figure 1 presents the symbol of the four standard floating point operators used in this work.



(a) Common (b) Conditionnal

Figure 2: Exact sum

Theorem 3 (ClosestUlp in ClosestProp) *The difference between a floating point number x and its rounded to the nearest (even) value is bounded as follows*

$$\forall (n,e) \in \mathbb{F}_{\mathcal{B}} \quad \textit{such that} \quad (n,e) \equiv \circ(x) \quad \textit{then} \quad |x - \circ(x)| \leq \beta^e/2.$$

Even rounding provides that if $|x - \circ(x)| = \beta^e/2$ then n is even.

1.2 Exact operations

We used for our proofs of the exact operation a result first published in [52]. It was later presented in [25] for the radix 2 notation.

Theorem 4 (Sterbenz in Fprop) *Given two bounded floating point numbers x and y such that $\frac{y}{2} \leq x \leq 2y$, the rational $x-y$ can be represented by a bounded floating point number. This fact is true for any bounded representation with any radix.*

We immediately prove the next result from its negation.

Theorem 5 (plusClosestLowerBound in Closest2Plus) *Given two radix 2 bounded floating point numbers x and y such that $x + y \neq x \oplus y$, we know that $|x \oplus y| \geq \max(|x|, |y|)/2$.*

Møller first, then Knuth proposed the common exact sum of Figure 2-(a) involving 4 floating point additions and 2 floating point subtractions on an IEEE compliant computer. The result is a pair (a', b') such that $a' = a \oplus b$ and $a' + b' = a + b$. We did not validated this long published algorithm, but we set up its specification and we deduced the following properties on its outputs.

Theorem 6 (errorBoundedPlus in ClosestPlus) *Given two bounded floating point numbers a and b, we define $a' \in \mathbb{F}_{\mathcal{B}}$ such that $a' = a \oplus b$. Provided that a' does not overflow, we can define $b' \in \mathbb{F}_{\mathcal{B}}$ such that $a' + b' = a + b$.*

The rounded sum $a \oplus b$ and the additional error $a + b - (a \oplus b)$ are both integer multiple of β^l where l is the lowest of the ulps of a and b. No significant bit is created right of the rightmost non zero significant bit of the inputs and the ulp of the sum is limited as we can see in the next fact.

Theorem 7 (plusExpBound in FroundPlus and errorBoundedPlus in ClosestPlus)
For any bounded representation (n_a, e_a) and (n_b, e_b) of a and b, we can define (n, e), a bounded representation of $a \oplus b$, and (n', e'), a bounded representation of $a + b - (a \oplus b)$, such that

$$\min(e_a, e_b) = e' \leq e \leq \max(e_a, e_b) + 1.$$

This last Theorem prompts the fact that the addition is accurate up to one unit in the last place. Such result is needed to show that the numerical tools of the next Section actually do some work.

Theorem 8 (plusErrorBound1 in ClosestPlus and plusErrorBound2 in Closest2Plus) *Given two floating point numbers a and b, we know that*

$$|a+b-(a\oplus b)| < \frac{|a\oplus b|}{2}\frac{\beta}{n_{max}} \leq \max(|a|,|b|)\frac{\beta}{n_{max}}$$

It has also been proved by Møller and presented in Knuth's second edition that one can get the correct pair (a',b') by an early exit of the exact sum provided $|b| \leq |a|$. The conditional sum of Figure 2-(b) returns the exact pair $(a',b'') = (a',b')$ with only 2 floating point additions and one floating point subtraction.

Knuth suggested that the early exit is still valid if a and b are both canonical and share the same exponent. We prove here a result rephrased from [13] that a and b just have to be bounded.

Theorem 9 (ExtDekker in EFast2Sum) *On a radix 2 IEEE inspired floating point notation, given two bounded floating point numbers $a \equiv (n_a, e_a)$ and $b \equiv (n_b, e_b)$, the conditional sum presented Figure 2-(b) returns the exact pair $(a\oplus b, a+b-a\oplus b)$ provided $e_b \leq e_a$.*

An exact multiplication is also available. It computes a pair (a',b') such that $a' = a\otimes b$ and $a'+b' = a\times b$ with 7 floating point multiplications, 5 floating point additions and 5 floating point subtractions. These operators are surveyed in [13]. Here are the properties obtained from the desired specification of a' and b', not from the algorithm. The condition on the exponent is not necessary but it is sufficient to discard cases of underflow.

Theorem 10 (errorBoundedMult in FroundMult) *Given the floating point numbers a and b, we define $a' \in \mathbb{F}_{\mathcal{B}}$ such that $a' = a\otimes b$. Let (n_a, e_a) and (n_b, e_b) be two bounded representations of a and b, provided that a' does not overflow and $e_a + e_b \geq -e_{min}$, we can define $b' \in \mathbb{F}_{\mathcal{B}}$ such that $a'+b' = a\times b$.*

The rounded product $a\otimes b$ and the additional error $a\times b - a\otimes b$ are both integer multiple of β^s where s is the sum of the ulps of a and b. The ulp of the product is also limited.

Theorem 11 (multExpUpperBound, multExpMin and multErrorExpMin in FroundMult)
For any possible bounded representation (n_a, e_a) and (n_b, e_b) of a and b, provided $e_a + e_b \geq -e_{min}$, we can define (n,e) bounded representation of $a\otimes b$ and (n',e') bounded representation of $a\times b - a\otimes b$ such that

$$e_a + e_b = e' \leq e \leq e_a + e_b + m_{\mathcal{R}} + 1.$$

It was proved that comparable error quantities b', that always fit in a common floating point number, may be defined for the division and square root [8].

1.3 Expansions

An **expansion** is defined as a finite sequence $\mathbf{x} = (x_{n-1}, \cdots, x_1, x_0)$ of floating point numbers. The value represented is the exact, not rounded, sum of its components $\sum x_i$. The length of the expansion is the number of its components.

We say that $\widehat{x}$ is a **fair most significant component** of $\mathbf{x}$ if $\widehat{x}$ equals to the rounded to the nearest value of x' with $|x' - \sum x_i| \leq \beta^e/n_{max}$ and (n,e) is the canonical representation of $\widehat{x}$. This includes the case where $\widehat{x}$ equals the rounded to the nearest value of the sum of the component of $\mathbf{x}$.

Any component of an expansion may be equal to zero, but the subsequence of the non-zero components x_i must be non overlapping and ordered by magnitude. The non overlapping condition means that two components cannot have significant bits with the same weight that is specified for each $i > 0$ as

$$\exists (n,e) \in \mathbb{F}_{\mathcal{R}} \quad \text{such that} \quad (n,e) \equiv x_i \quad \text{and} \quad |x_{i-1}| < \beta^e.$$

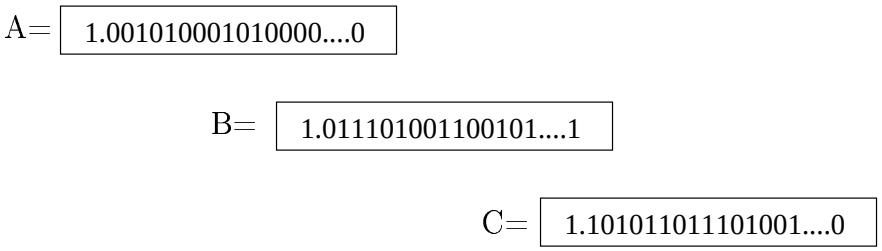


Figure 3: Floating point numbers A and B do not overlap whereas B and C do.

A **non-adjacent expansion** is an expansion **x** where the non-zero subsequence has non-adjacent components. The condition is specified below. There is at least one bit set to 0 between the two nearest non-zero (signed) bits of two consecutive components.

$$\exists (n,e) \in \mathbb{F}_{\mathcal{R}} \quad \text{such that} \quad (n,e) \equiv x_i \quad \text{and} \quad |x_{i-1}| < \beta^e/2$$

We have already presented two examples in [13] showing that any floating point pair (n, e) where $e \geq -e_{min}$ can be expressed as an expansion or an non-adjacent expansion.

As noted in the introduction, previous work by Priest, Shewchuck and ourselves has produced arithmetic operators on expansions and useful primitives for computational geometry. In this process, we recognized that the operators for the addition and the multiplication computing the least significant digits first do not produce length optimal results but tend to break the expansion in a large number of small components.

It is possible to compute the arithmetic operations most significant digits first. The division operator computes the components of the quotient like this. As a drawback, some of the components may overlap slightly. Yet, the sum of any two following components $x_i \oplus x_{i-1}$) is a fair most significant component for the remaining of the pseudo-expansion $\sum_{j \leq i} x_j$. In our former work [15], the sequence of the quotient digits is cleaned by a routine that produces an actual expansion.

We define **pseudo-expansions** as slightly overlapping expansions. The division of pseudo-expansion is copied from the existing one except that it does not require the final cleaning loops. The addition and the multiplication are inspired from the on-line operator to produce the result most significant digit first as components are available and safe. As we do for on-line arithmetic, we recognize that for the three operators, least significant digits are not necessary at the beginning of the process and the operators are transformed from most significant digit first operators that require its inputs to be exactly known to adaptable operators that need only a limited number of most significant components to deliver the first components of the output.

2 Toolset for the addition, the multiplication and the division

The drawings of this Section represent actual code implementations in C language for the operators. We can see that only 5 routines are used (MQ, PP, PQ, Σ_3 and Σ_5). Among these routines, the numerical behavior of only Σ_3 and Σ_5 are difficult to analyze as the three other ones only sort and produce components as they are needed.

2.1 Addition

Figure 4 represents the algorithm implemented to compute the sum of pseudo-expansions A and B. We use two primitive operators MQ (merge queue) and Σ_3 (insert and sum of Figure 5). Arrows represent streams of floating point number flowing most significant digit first. The components of the result are in the stream named C in Figure 4. The output is only a pseudo-expansion since the a' are sorted by decreasing order of magnitude but they may partially overlap each other. However, two components separated by at least one non-zero component never overlap.

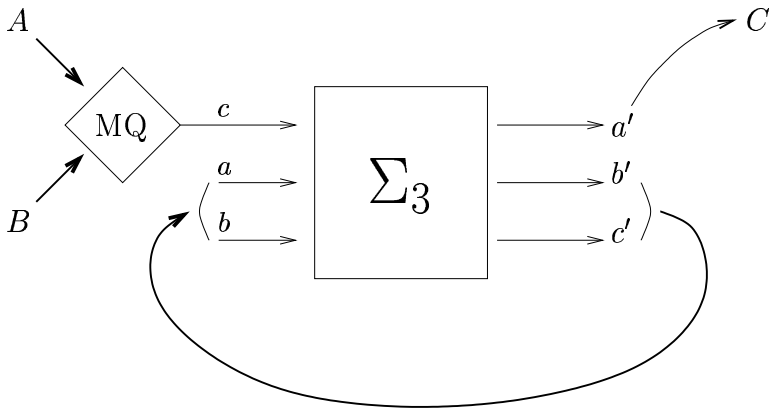


Figure 4: Representation of the addition. Case where $c' \neq 0$.

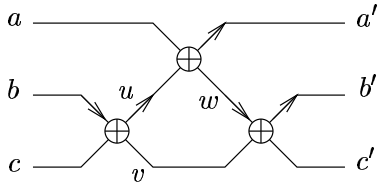


Figure 5: Definition of the Σ_3 operator.

We deduce the correct behavior of the operation by induction from the following Theorem. The property on the input floating point number c is inherited from the fact that the two input streams A and B are pseudo-expansion merged by the MQ operator by order of magnitude.

Theorem 12 (bound3Sum and exp3Sum in ThreeSum2) *Let a, b and c be the input of the Σ_3 operator (see Figure 5) and let (n_a, e_a), (n_b, e_b) and (n_c, e_c) be bounded representations of a, b and c. Provided that a, b or c is zero or $e_a \geq e_b \geq e_c$, the Σ_3 operator where the first and the last exact additions use the early exit, returns three numbers a', b', c' represented by (n'_a, e'_a), (n'_b, e'_b) and (n'_c, e'_c) such that $a + b + c = a' + b' + c'$ and $e'_a \geq e'_b \geq e'_c \geq e_c$ and finaly either $c' = 0$ or*

$$|b' + c'| < \frac{3\beta}{n_{max}}|a'|.$$

If c' equals to zero, a' and b' are not relevant enough. They remain in the operator as a and b for the next iteration. On the opposite, if c' is not equal to zero, a' is relevant. It is one component of the result, b' and c' are kept in the operator for the next iteration.

2.2 Multiplication

The multiplication of expansions or pseudo-expansions of A (size n_A) by B (size n_B) generates the $n_A \times n_B$ partial products and computes their sum. Figure 6 represents the algorithm implemented. We use new and extended primitives : PP, a partial product generator, PQ, a priority queue extending the MQ, and Σ_5 a 5 entries insert-sum of Figure 8.

Numbers flow by pair since $A_i \times B_j$ produces two floating point numbers: $(A_i \times B_j)_{High}$ that is inserted in c and $(A_i \times B_j)_{Low}$ inserted in e. We easily see that Σ_3 is a simplified instantiation of Σ_5 where the last 2 inputs are zeros. A formal extension of Theorem 12 will guarantee that this operator produces the desired output.

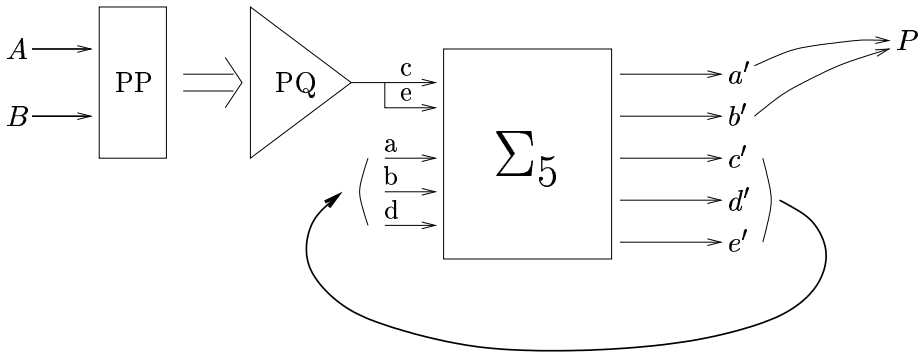


Figure 6: Representation of the multiplication - case where $e' \neq 0$.

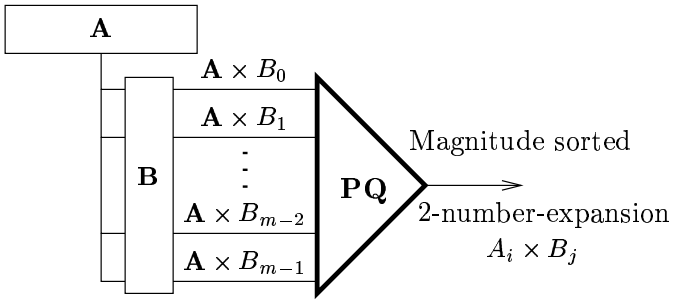


Figure 7: Generation of partial product and priority queue.

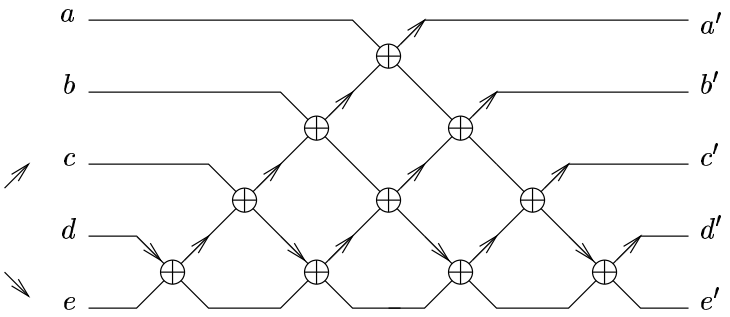


Figure 8: Definition of the Σ_5 operator.

If d' equals to zero, then $e' = 0$ and a', b' and c' are not relevant enough. They remain in the operator for the next iteration. If $d' \neq 0$, a' is relevant. It is one component of the result. If e' is also non zero, b' is immediately used as the next component of the result. The three remaining floating point numbers are kept in the operator for the next iteration.

2.3 Division

Figure 9 represents the division of two pseudo-expansions R by D. At iteration i where $d' \neq 0$, a new approximate quotient digit q_i is guessed from a fair most significant component d_l of the divisor $D = (d_l \cdots d_0)$ and a fair most significant component a' of the remainder R_i: $q_i = \circ(a'/d_l)$. Since R_i is at least divided by

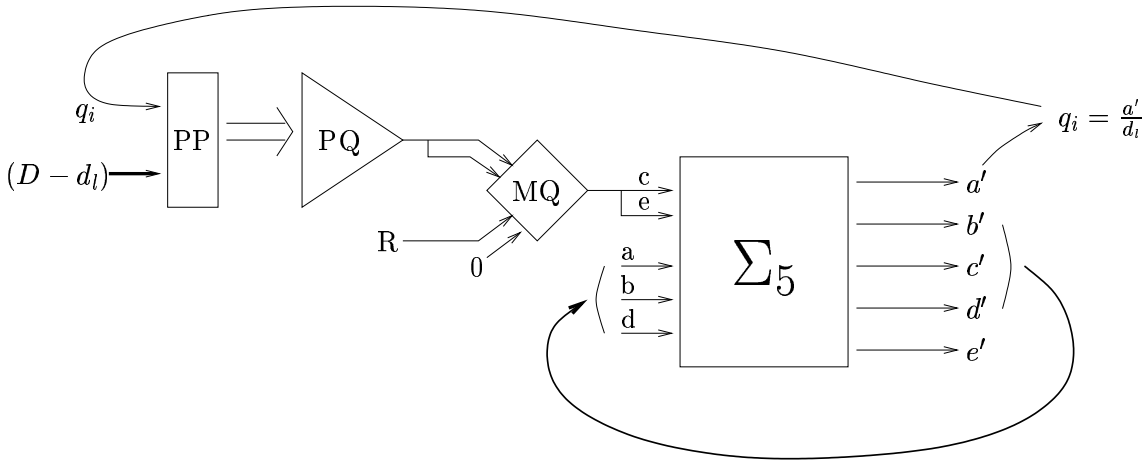


Figure 9: Representation of the non-restoring division - case where b' is relevant.

a constant factor at each iteration, the algorithm is converging.

$$\frac{R_0}{D} = \sum_{i=0}^{n-1} q_i + \frac{R_n}{D}$$

We use four primitive operators. The partial product PP, and the priority queue PQ, sort by magnitude the pairs generated by the product of q_i by $D - d_l$. The merging queue MQ, takes the most significant numbers between the components of the numerator that have not been used so far and the numbers given by the priority queue. The Σ_5 operator adds the new component to the left-over components. The number a' is significant enough when $d' \neq 0$. This process guarantees that a' is a relevant approximation of R_i. We will prove that $e' = 0$.

In modern computers, the floating point division is a very slow operator. For example, Intel Pentium processors need 39 cycles [29] whereas multiplications and additions may have a throughput of 1 cycle. With prescaled (non-restoring) division, we reduce the number of floating point division to 1. As a price, we have to exactly prescale the numerator R involving n_R multiplications. As a matter of fact, the prescaling fits the size of the second input of the MQ operator that is waiting for a pair of floating point digits from the PQ and now also from the numerator.

Conclusion

We have presented a new adaptable numeric core inspired both from floating point expansions and from on-line arithmetic. Our choice was to present only the arithmetic core as we have presented in past publications how a core can be used efficiently to compute matrix determinants for example [15]. Building a general purpose runtime environment raises question in areas like parallelism (demand driven *vs.* dataflow...) and in compiler techniques (efficiency of object oriented code generated by existing compilers...) among others.

The numeric core is cut down to five tools. The Σ_3 operator that contains many arithmetic operations is proved correct. We only check by hand the induction on time as the digits flow into the system. The Σ_5 operator should soon also be formally proved.

The proofs have been validated by the Coq assistant. Developing the proofs, we have formally proved many result long published in the literature and we have extended a few of them. This work may let users

- Develop application specific adaptable libraries based on the toolset.
- Easily write new formal proofs based on the growing set of sensible validated facts.

The set of the validated facts is now sufficient to shield the user from the difficult implementation details of floating point arithmetic. The proof of the key Theorem on the Σ_3 operator does not use any low level result. It just uses the many lower and upper bounds defined in the Theorems of Section 1.

References

[1] Algirdas Avižienis. Signed digit number representations for fast parallel arithmetic. *IRE Transactions on Electronic Computers*, 10:389–400, 1961.

[2] Francis Avnaim, Jean-Daniel Boissonnat, Olivier Devillers, Franco Preparata, and Mariette Yvinec. Evaluating signs of determinants using single precision arithmetic. *Algorithmica*, 17(2):111–132, 1997.

[3] David H. Bailey. Algorithm 719, multiprecision translation and execution of fortran programs. *ACM Transactions on Mathematical Software*, 19(3):288–319, 1993.

[4] David H. Bailey. A fortran-90 based multiprecision system. Technical report RNR-94-013, NASA Ames Research Center, 1994.

[5] Jesse L. Barlow. Error analysis of update methods for the symmetric eigenvalue problem. *SIAM Journal of Matrix Analysis and Applications*, 14(2):598–618, 1993.

[6] David Berthelot and Marc Daumas. Computing on sequences of embedded intervals. *Reliable Computing*, 3(3):219–227, 1997.

[7] Hans-Juergen Boehm. Constructive real interpretation of numerical programs. *ACM SIGPLAN Notices*, 22(7):214–221, 1987.

[8] Gerd Bohlender, Wolfgang Walter, Peter Kornerup, and David W. Matula. Semantics for exact floating point operations. In Peter Kornerup and David W. Matula, editors, *Proceedings of the 10th Symposium on Computer Arithmetic*, pages 22–26, Grenoble, France, 1991. IEEE Computer Society Press.

[9] Jean-Daniel Boissonnat and Franco P. Preparata. Robust plane sweep for intersecting segments. *SIAM Journal on Computing*, 29(5):1401–1421, 2000.

[10] Richard P. Brent. A fortran multiple-precision arithmetic package. *ACM Transactions on Mathematical Software*, 4(1):57–70, 1978.

[11] C. Burnikel, R. Fleischer, K. Mehlhorn, and S. Schirra. Efficient exact geometric computation made easy. In *Proceedings of the 15th Annual ACM Symposium on Computational Geometry*, pages 341–350, Miami, Florida, 1999.

[12] G. Corbaz, Jean Duprat, Bertrand Hochet, and Jean-Michel Muller. Implementation of a VLSI polynomial evaluator for real-time applications. In *International Conference on Application-Specific Array Processors*, 1991.

[13] Marc Daumas. Multiplications of floating point expansions. In Israel Koren and Peter Kornerup, editors, *Proceedings of the 14th Symposium on Computer Arithmetic*, pages 250–257, Adelaide, Australia, 1999.

[14] Marc Daumas and Claire Finot. Division of floating point expansions. In *IMACS-GAMMM International Symposium on Scientific Computing, Computer Arithmetic and Validated Numerics*, pages 45–46, Budapest, Hungaria, 1998.

[15] Marc Daumas and Claire Finot. Division of floating point expansions with an application to the computation of a determinant. *Journal of Universal Computer Science*, 5(6):323–338, 1999.

[16] Marc Daumas, Jean-Michel Muller, and Arnaud Tisserand. Very high radix on-line arithmetic for accurate computations. In *Proceedings of the 15th IMACS World Congress on Computational and Applied Mathematics*, volume 2, pages 347–352, Berlin, Germany, 1997.

[17] Marc Daumas, Jean-Michel Muller, and Jean Vuillemin. Implementing on-line arithmetic on PAM. In Reiner W. Hartenstein and Michael Z. Servít, editors, *Field-Programmable Logic: Architectures Synthesis and Applications*, pages 196–207, Prague, Czech Republic, 1994. Springer Verlag.

[18] T. J. Dekker. A floating point technique for extending the available precision. *Numerische Mathematik*, 18(3):224–242, 1971.

[19] Olivier Devillers and Franco Preparata. A probabilistic analysis of the power of arithmetic filters. *Discrete and Computational Geometry*, 20:523–547, 1998.

[20] Jack J. Dongarra. Performance of various computers using standard linear equations software. *Computer architecture news*, 20(3):22–44, 1992.

[21] Jack J. Dongarra. Performance of various computers using standard linear equations software. Report CS-89-85, University of Tennessee, 1999.

[22] Steven Fortune and Christopher J. Van Wyk. Efficient exact arithmetic for computational geometry. In *Proceedings of the 9th ACM Symposium on Computational Geometry*, pages 163–172, 1993.

[23] Christiane Frougny. Representation of numbers in non classical numeration systems. In Peter Kornerup and David W. Matula, editors, *Proceedings of the 10th Symposium on Computer Arithmetic*, pages 17–21, Grenoble, France, 1991. IEEE Computer Society Press.

[24] Brice Goglin. Calcul sur des expansions de taille fixe. Technical report, Ecole Normale Supérieure de Lyon, 2000.

[25] David Goldberg. What every computer scientist should know about floating point arithmetic. *ACM Computing Surveys*, 23(1):5–47, 1991.

[26] Tobjörn Granlund. *The GNU multiple precision arithmetic library*, 2000. Version 3.1.

[27] Nicholas J. Higham. *Accuracy and stability of numerical algorithms.* SIAM, 1996.

[28] Gérard Huet, Gilles Kahn, and Christine Paulin-Mohring. The Coq Proof Assistant: A Tutorial: Version 6.1. Technical Report 204, Institut National de Recherche en Informatique et en Automatique, Le Chesnay, France, 1997.

[29] Intel. *Pentium Family: Developer's Manual*, 1995. Pentium Processor.

[30] William Kahan. Further remarks on reducing truncation errors. *Communications of the ACM*, 8(1):40, 1965.

[31] V. Karamcheti, C. Li, I. Pechtchanski, and Chee Yap. A core library for robust numeric and geometric computation. In *Proceedings of the 15th Annual ACM Symposium on Computational Geometry*, pages 351–359, Miami, Florida, 1999.

[32] Donald E. Knuth. *The art of computer programming: Seminumerical Algorithms.* Addison Wesley, 1969.

[33] Donald E. Knuth. *The art of computer programming: Seminumerical Algorithms.* Addison Wesley, 1973.

[34] Marianne E. Louie and Miloš D. Ercegovac. On digit recurrence division implementation for field programmable gate arrays. In Earl Swartzlander, Mary Jane Irwin, and Graham Jullien, editors, *Proceedings of the 11th Symposium on Computer Artithmetic*, pages 202–209, Windsor, Ontario, 1993. IEEE Computer Society Press.

[35] David W. Matula. Basic digit sets for radix representation. *Journal of the ACM*, 29(4):1131–1143, 1982.

[36] Christophe Mazenc. On the redundancy of real number representation systems. Research report 93-16, Laboratoire de l'Informatique du Parallélisme, Lyon, France, 1993.

[37] Christophe Mazenc and Xavier Merrheim. Handling numbers in high precision and controlling the numerical precision in a serial system on a parallel MIMD machine. In *26th International Conference on System Sciences*, pages 1196–1198, Honolulu, Hawaii, 1993.

[38] Valérie Ménissier. *Arithmétique Exacte : Conception, Algorithmique et Performances d'une Implantation Informatique en Précision Arbitraire.* PhD thesis, Université Paris VI, Paris, France, 1994.

[39] Valérie Ménissier-Morain. Conception et algorithmique d'une représentation d'arithmétique réelle en précision arbitraire. In *Real Numbers and Computers - Les nombres réels et l'ordinateur (St-Étienne, France)*, pages 47–64, 1995.

[40] Dominique Michelucci and Jean-Michel Moreau. Lazy arithmetic. *IEEE Transactions on Computers*, 46(9):961–975, 1997.

[41] Ole Møller. Note on quasi double-precision. *BIT*, 5(4):251–255, 1965.

[42] Ole Møller. Quasi double-precision in floating point addition. *BIT*, 5(1):37–50, 1965.

[43] Jean-Michel Muller. Some characterizations of functions computable in on-line arithmetic. *IEEE Transactions on Computers*, 43(6):752–755, 1994.

[44] Asger Munk Nielsen and Peter Kornerup. Redundant radix representations of rings. *IEEE Transactions on Computers*, 48(11):1153–1165, 1999.

[45] Michèle Pichat. Correction d'une somme en arithmétique à virgule flottante. *Numerische Mathematik*, 19:400–406, 1972.

[46] Douglas M. Priest. Algorithms for arbitrary precision floating point arithmetic. In Peter Kornerup and David Matula, editors, *Proceedings of the 10th Symposium on Computer Arithmetic*, pages 132–144, Grenoble, France, 1991. IEEE Computer Society Press.

[47] Peter-Michael Seidel. Exact arithmetic based on floating-point numbers. In *IMACS-GAMMM International Symposium on Scientific Computing, Computer Arithmetic and Validated Numerics*, page 123, Karlsruhe, Germany, 2000.

[48] Bernard Serpette, Jean Vuillemin, and Jean-Claude Hervé. BigNum: a portable and efficient package for arbitrary-precision arithmetic. Technical Report 2, Digital Equipment Corporation, Paris Research Laboratory, 1989.

[49] Jonathan R. Shewchuk. Robust adaptative floating point geometric predicates. In *Proceedings of the 12th Annual ACM Symposium on Computational Geometry*, pages 141–150, Philadelphia, Pensylvania, 1996.

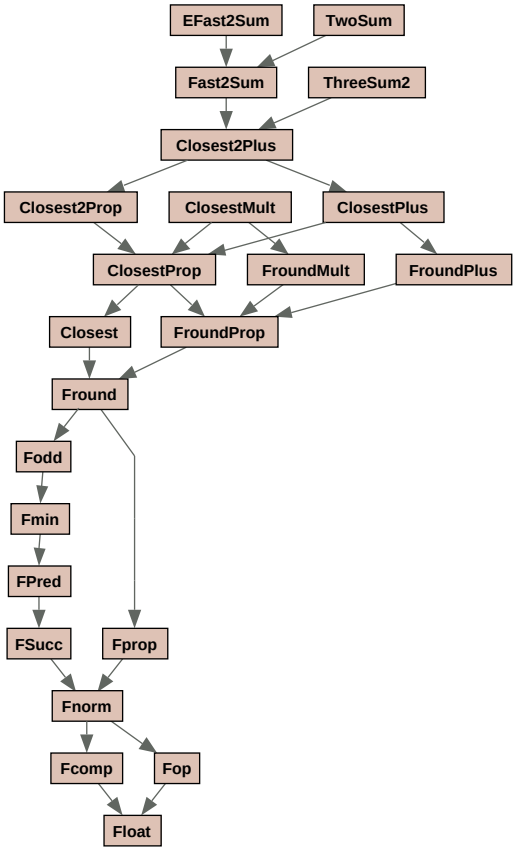
[50] Jonathan R. Shewchuk. Adaptive precision floating-point arithmetic and fast robust geometric predicates. In *Discrete and Computational Geometry*, volume 18, pages 305–363, 1997.

[51] Danny C. Sørensen and Ping Tak Peter Tang. On the orthogonality of eigenvectors computed by divide and conquer methods techniques. *SIAM Journal of Numerical Analysis*, 28(6):1752–1775, 1991.

[52] Pat H. Sterbenz. *Floating point computation.* Prentice Hall, 1974.

[53] Kishor S. Trivedi and Miloš D. Ercegovac. On-line algorithms for division and multiplication. *IEEE Transactions on Computers*, 26(7):681–687, 1977.

A Hierarchy



B File Float

Record *float*: *Set* := *Float* {

Fnum:Z;

Fexp:Z
}

In the following, we write $(Float\ x\ y)$ as $\{x, y\}$ and $(Fnum\ f)$ and $(Fexp\ f)$ as $n[f]$ and $e[f]$.

Theorem $floatEq$: $\forall p, q$: $float.\ n[p] = n[q] \Rightarrow e[p] = e[q] \Rightarrow p = q$.

Theorem $floatDec$: $\forall x, y$: $float.\ \{x = y\} + \{\neg(x = y)\}$.

Definition $Fzero := \lambda x$: $Z.\ \{0, x\}$.

Definition $is\_Fzero := \lambda x$: $float.\ n[x] = 0$.

In the following we write $(is\_Fzero\ f)$ as $f = 0$.

Theorem $is\_Fzero\_or\_not$: $\forall x$: $float.\ x = 0 \vee \neg x = 0$.

Definition $FtoR := \lambda x$: $float.\ n[x] * radix^{e[x]}$.

In the following we write $<R>\ (FtoR\ f_1) == (FtoR\ f_2)$ as $<R>\ f_1 == f_2$.

Theorem is_Fzero_rep1: $\forall x$: $float.\ x = 0 \Rightarrow <R>\ x == 0$.

Theorem is_Fzero_rep2: $\forall x$: $float.\ <R>\ x == 0 \Rightarrow x = 0$.

Theorem $NisFzeroComp$: $\forall x, y$: $float.\ \neg x = 0 \Rightarrow <R>\ x == y \Rightarrow \neg y = 0$.

Theorem $FtoREqInv1$: $\forall p, q$: $float.$
$\neg p = 0 \Rightarrow <R>\ p == q \Rightarrow n[p] = n[q] \Rightarrow p = q$.

Theorem $FtoREqInv2$: $\forall p, q$: $float.\ <R>\ p == q \Rightarrow e[p] = e[q] \Rightarrow p = q$.

Definition $Fdigit := \lambda p$: $float.\ digit(|n[p]|)$.

In the following we write $(Fdigit\ f)$ as $digit(f)$.

Definition $Fshift := \lambda n$: $nat.\ \lambda x$: $float.\ \{n[x] * radix^n, e[x] - n\}$.

In the following we write $(Fshift\ n\ f)$ as $shift(n, f)$.

Theorem $sameExpEq$: $\forall p, q$: $float.\ <R>\ p == q \Rightarrow e[p] = e[q] \Rightarrow p = q$.

Theorem $FshiftFdigit$: $\forall n$: $nat.\ \forall x$: $float.$
$\neg x = 0 \Rightarrow digit(shift(n, x)) = digit(x) + n$.

Theorem $FshiftCorrect$: $\forall n$: $nat.\ \forall x$: $float.$
$<R>\ shift(n, x) == x$.

Theorem $FshiftCorrectInv$: $\forall x, y$: $float.$
$<R>\ x == y \Rightarrow e[x] \leq e[y] \Rightarrow shift(|e[y] - e[x]|, y) = x$.

Theorem $Fshift0$: $\forall x$: $float.\ shift(0, x) = x$.

Theorem $FshiftCorrectSym$: $\forall x, y$: $float.$
$<R>\ x == y \Rightarrow \exists n$: $nat.\ \exists m$: $nat.\ shift(n, x) = shift(m, y)$.

Theorem $FshiftAdd$: $\forall n, m$: $nat.\ \forall p$: $float.$
$shift(n + m, p) = shift(n, shift(m, p))$.

Theorem $ReqGivesEqwithSameExp$: $\forall p, q$: $float.$
$\exists r$: $float.\ \exists s$: $float.\ <R>\ p == r \wedge <R>\ q == s \wedge e[r] = e[s]$.

Theorem $FdigitEq$: $\forall x, y$: $float.$
$\neg x = 0 \Rightarrow <R>\ x == y \Rightarrow digit(x) = digit(y) \Rightarrow x = y$.

C File Fop

Definition $Fplus := \lambda x, y\colon float.$
$\{n[x] * (radix^{|e[x]-min(e[x],e[y])|}) + n[y] * (radix^{|e[y]-min(e[x],e[y])|}),$
$min(e[x], e[y])\}.$

In the following, we write $(Fplus\ f_1\ f_2)$ as $f_1 + f_2$

Theorem $Fplus\_correct\colon \forall x, y\colon float. <R> x + y == x + y.$

Definition $Fop := \lambda x\colon float. \{-n[x], e[x]\}.$

In the following, we write $(Fop\ f)$ as $-f$

Theorem $Fopp\_correct\colon \forall x\colon float. <R>\ -x == -x.$

Theorem $Fopp\_Fopp\colon \forall p\colon float.\ --p = p.$

Definition $Fabs := \lambda x\colon float. \{|n[x]|, e[x]\}.$

In the following, we write $(Fabs\ f)$ as $|f|$

Theorem $Fabs\_correct1\colon \forall x\colon float. 0 \leq (x) \Rightarrow <R> |x| == x.$

Theorem $Fabs\_correct2\colon \forall x\colon float. (x) \leq 0 \Rightarrow <R> |x| == -x.$

Theorem $Fabs\_correct\colon \forall x\colon float. <R> |x| == |x|.$

Theorem $Fabs\_Fzero\colon \forall x\colon float.\ x = O \Rightarrow\ |x| = O.$

Theorem $Fdigit\_abs\colon \forall x\colon float. digit(|x|) = digit(x).$

Definition $Fminus := \lambda x, y\colon float. x + -y.$

In the following, we write $(Fminus\ f_1\ f_2)$ as $f_1 - f_2$

Theorem $Fminus\_correct\colon \forall x, y\colon float. <R> x - y == x - y.$

Theorem $Fopp\_Fminus\colon \forall p, q\colon float.\ -(p - q) = q - p.$

Theorem $Fopp\_Fminus\_dist\colon \forall p, q\colon float.\ -(p - q) = -p - (-q).$

Theorem $minusSameExp\colon \forall x, y\colon float. e[x] = e[y] \Rightarrow x - y = \{n[x] - n[y], e[x]\}.$

Definition $Fmult := \lambda x, y\colon float. \{n[x] * n[y], e[x] + e[y]\}.$

In the following, we write $(Fmult\ f_1\ f_2)$ as $f_1 * f_2$

Theorem $Fmult\_correct\colon \forall x, y\colon float. <R> x * y == x * y.$

D File Fcomp

Lemma $Fle\_Zle\colon \quad \forall n1, n2, d\colon Z. n1 \leq n2 \Rightarrow \{n1, d\} \leq \{n2, d\}.$

Lemma $Flt\_Zlt\colon \quad \forall n1, n2, d\colon Z. n1 < n2 \Rightarrow \{n1, d\} < \{n2, d\}.$

Lemma $Fle\_Fge\colon \quad \forall x, y\colon float. x \leq y \Rightarrow y \geq x.$

Lemma $Fge\_Zge\colon \quad \forall n1, n2, d\colon Z. n1 \geq n2 \Rightarrow \{n1, d\} \geq \{n2, d\}.$

Lemma $Flt\_Fgt\colon \quad \forall x, y\colon float. x < y \Rightarrow y > x.$

Lemma $Fgt\_Zgt\colon \quad \forall n1, n2, d\colon Z. n1 > n2 \Rightarrow (\{n1, d\}) > (\{n2, d\}).$

Lemma $Fle\_refl\colon \quad \forall x, y\colon float. x == y \Rightarrow x \leq y.$

Lemma $Fle\_trans\colon \quad \forall x, y, z\colon float. x \leq y \Rightarrow y \leq z \Rightarrow x \leq z.$

Theorem $Rlt\_Fexp\_eq\_Zlt\colon \forall x, y\colon float. x < y \Rightarrow e[x] = e[y] \Rightarrow n[x] < n[y].$

Theorem $Rle\_Fexp\_eq\_Zle\colon \forall x, y\colon float. x \leq y \Rightarrow e[x] = e[y] \Rightarrow n[x] \leq n[y].$

Theorem $LtR0Fnum\colon \forall p\colon float. 0 < p \Rightarrow 0 < n[p].$

Theorem $LeR0Fnum\colon \forall p\colon float. 0 \leq p \Rightarrow 0 \leq n[p].$

Theorem $LeFnum0\colon \forall x\colon float. 0 \leq n[x] \Rightarrow 0 \leq x.$

Theorem $R0LtFnum$: $\forall p$: $float$. $p < 0 \Rightarrow n[p] < 0$.

Theorem $R0LeFnum$: $\forall p$: $float$. $p \leq 0 \Rightarrow n[p] \leq 0$.

Theorem $Le0Fnum$: $\forall x$: $float$. $n[x] \leq 0 \Rightarrow x \leq 0$.

E File Fnorm

Record $Fbound$: Set := $Bound$ {

$vNum$:nat;

$dExp$:nat
}

In the following, we write ($vNum$ b) and ($dExp$ b) as $v[b]$ and $d[b]$.

Definition $Fbounded$:= λb: $Fbound$. λd: $float$.
$-v[b] \leq n[d] \wedge n[d] \leq v[b] \wedge -d[b] \leq e[d]$.

In the following we write ($Fbounded$ f) as $B[p]$.

Theorem $isBounded$: $\forall b$: $Fbound$. $\forall p$: $float$. $\{B[p]\} + \{\neg(B[p])\}$.

Theorem $pBounded\_absolu$: $\forall b$: $Fbound$. $\forall p$: $float$. $B[p] \Rightarrow |n[p]| \leq v[b]$.

Theorem $FzeroisZero$: $\forall b$: $Fbound$. $<R>$ ($Fzero$ $- d[b]$) == 0.

Theorem $FboundedFzero$: $\forall b$: $Fbound$. $B[(Fzero\ - d[b])]$.

Theorem $oppBounded$: $\forall b$: $Fbound$. $\forall x$: $float$. $B[x] \Rightarrow B[-x]$.

Theorem $oppBoundedInv$: $\forall b$: $Fbound$. $\forall x$: $float$. $B[-x] \Rightarrow B[x]$.

Theorem $absFBounded$: $\forall b$: $Fbound$. $\forall f$: $float$. $B[f] \Rightarrow B[|f|]$.

Theorem $FboundedEqExp$: $\forall b$: $Fbound$. $\forall p, q$: $float$. $B[p] \Rightarrow$
$<R>\ p == q \Rightarrow e[p] \leq e[q] \Rightarrow B[q]$.

Theorem $eqExpLess$: $\forall b$: $Fbound$. $\forall p, q$: $float$. $B[p] \Rightarrow$
$<R>\ p == q \Rightarrow \exists r$: $float$. $B[r] \wedge <R>\ r == q \wedge e[q] \leq e[r]$.

Theorem $FboundAlt$: $\forall b$: $Fbound$. $\forall p$: $float$. $|n[p]| \leq v[b] \Rightarrow -d[b] \leq e[p] \Rightarrow B[p]$.

Theorem $eqExpMax$: $\forall b$: $Fbound$. $\forall p, q$: $float$. $B[p] \Rightarrow B[q] \Rightarrow$
$|p| \leq q \Rightarrow \exists r$: $float$. $B[r] \wedge <R>\ r == p \wedge e[r] \leq e[q]$.

Theorem $maxFbounded$: $\forall b$: $Fbound$. $\forall z$: Z. $- d[b] \leq z \Rightarrow B[\{v[b], z\}]$.

Theorem $maxMax$: $\forall b$: $Fbound$. $\forall p$: $float$. $\forall z$: Z. $B[p] \Rightarrow$
$e[p] \leq z \Rightarrow |p| \leq \{v[b], z\}$.

Hypothesis $pGivesBound$: $v[b] = radix^{precision} - 1$.

Theorem lt_vNum: $O < v[b]$.

Theorem $digitVNumiSPrecision$: $digit(v[b]) = precision$.

Theorem $pGivesDigit$: $\forall p$: $float$. $B[p] \Rightarrow digit(p) \leq precision$.

Theorem $digitGivesBoundedNum$: $\forall p$: $float$.
$digit(p) \leq precision \Rightarrow -v[b] \leq n[p] \wedge n[p] \leq v[b]$.

Definition $Fnormal$:= λp: $float$. $B[p] \wedge digit(p) = precision$.

In the following we write ($Fnormal$ f) as $N[p]$.

Theorem $FnormalNotZero$: $\forall p$: $float$. $N[p] \Rightarrow \neg p = O$.

Theorem $FnormalUnique$: $\forall p, q$: $float$. $N[p] \Rightarrow N[q] \Rightarrow$
$<R>\ p == q \Rightarrow p = q$.

Theorem $FnormalFop$: $\forall p$: $float$. $N[p] \Rightarrow N[-p]$.

Theorem $FnormalLtPos$: $\forall p, q$: $float$. $N[p] \Rightarrow N[q] \Rightarrow$

$0 \leq p \Rightarrow p < q \Rightarrow e[p] < e[q] \vee e[p] = e[q] \wedge n[p] < n[q]$.

Theorem $FnormalLtNeg$: $\forall p, q$: $float$. $N[p] \Rightarrow N[q] \Rightarrow$
$q \leq 0 \Rightarrow p < q \Rightarrow e[q] < e[p] \vee e[p] = e[q] \wedge n[p] < n[q]$.

Definition $nNormMin := radix^{precision-1}$.

Theorem $nNormNotZero$: $\neg(nNormMin = O)$.

Theorem $digitnNormMin$: $digit(nNormMin) = precision$.

Theorem $nNrMMimLevNum$: $nNormMin \leq v[b]$.

Definition $firstNormalPos := \{nNormMin, -d[b]\}$.

Theorem $firstNormalPosNormal$: $N[firstNormalPos]$.

Theorem $pNormal\_absolu\_min$: $\forall p$: $float$. $N[p] \Rightarrow nNormMin \leq |n[p]|$.

Theorem $maxMaxBis$: $\forall p$: $float$. $\forall z$: Z. $B[p] \Rightarrow$
$e[p] < z \Rightarrow |p| < (\{nNormMin, z\})$.

Theorem $FnormalLtFirstNormalPos$: $\forall p$: $float$. $N[p] \Rightarrow$
$0 \leq p \Rightarrow firstNormalPos \leq p$.

Theorem $FnormalLtFirstNormalNeg$: $\forall p$: $float$. $N[p] \Rightarrow$
$p \leq 0 \Rightarrow p \leq -firstNormalPos$.

Theorem $FnormalBoundAbs$: $\forall p$: $float$. $N[p] \Rightarrow \{v[b], e[p] - 1\} < |p|$.

Definition $Fsubnormal := \lambda p$: $float$. $B[p] \wedge e[p] = -d[b] \wedge digit(p) < precision$.

In the following we write $(Fsubnormal\ f)$ as $S[p]$.

Theorem $FsubnormFopp$: $\forall p$: $float$. $S[p] \Rightarrow S[-p]$.

Theorem $FsubnormalUnique$: $\forall p, q$: $float$. $S[p] \Rightarrow S[q] \Rightarrow <R>\ p == q \Rightarrow p = q$.

Theorem $FsubnormalLt$: $\forall p, q$: $float$. $S[p] \Rightarrow S[q] \Rightarrow$
$p < q \Rightarrow n[p] < n[q]$.

Theorem $LtFsubnormal$: $\forall p, q$: $float$. $S[p] \Rightarrow S[q] \Rightarrow$
$n[p] < n[q] \Rightarrow p < q$.

Theorem $pSubnormal\_absolu\_min$: $\forall p$: $float$. $S[p] \Rightarrow$
$|n[p]| < nNormMin$.

Theorem $FsubnormalLtFirstNormalPos$: $\forall p$: $float$. $S[p] \Rightarrow$
$0 \leq p \Rightarrow p < firstNormalPos$.

Theorem $FsubnormalnormalLtPos$: $\forall p, q$: $float$. $S[p] \Rightarrow N[q] \Rightarrow$
$0 \leq p \Rightarrow 0 \leq q \Rightarrow p < q$.

Theorem $FsubnormalnormalLtNeg$: $\forall p, q$: $float$. $S[p] \Rightarrow N[q] \Rightarrow$
$p \leq 0 \Rightarrow q \leq 0 \Rightarrow q < p$.

Definition $Fnormalize := \lambda p$: $float$.
Cases $n[p] = 0$? **of**
$true \Longrightarrow \{0, -d[b]\}$
$|\ false \Longrightarrow shift(min(precision - digit(p), |d[b] + e[p]|), p)$
end

In the following we write $(Fnormalize\ f)$ as $norm(p)$.

Theorem $FnormalizeCorrect$: $\forall p$: $float$. $<R>\ norm(p) == p$.

Theorem $Fnormalize\_Fopp$: $\forall p$: $float$. $norm(-p) = -norm(p)$.

Theorem $FnormalizeBounded$: $\forall p$: $float$. $B[p] \Rightarrow B[norm(p)]$.

Definition $Fcanonic := \lambda a$: $float$. $N[a] \vee S[a]$.

In the following we write $(Fcanonic\ f)$ as $C[p]$.

Theorem $FcanonicBound$: $\forall p{:}\ float.\ C[p] \Rightarrow B[p]$.

Theorem $pUCanonic\_absolu$: $\forall p{:}\ float.\ C[p] \Rightarrow |n[p]| \leq v[b]$.

Theorem $FcanonicFopp$: $\forall p{:}\ float.\ C[p] \Rightarrow C[-p]$.

Theorem $FcanonicFabs$: $\forall p{:}\ float.\ C[p] \Rightarrow C[|p|]$.

Theorem $FnormalizeCanonic$: $\forall p{:}\ float.\ B[p] \Rightarrow C[norm(p)]$.

Theorem $FcanonicLtPos$: $\forall p, q{:}\ float.\ C[p] \Rightarrow C[q] \Rightarrow$
$0 \leq p \Rightarrow p < q \Rightarrow e[p] < e[q] \vee e[p] = e[q] \wedge n[p] < n[q]$.

Theorem $FcanonicLtNeg$: $\forall p, q{:}\ float.\ C[p] \Rightarrow C[q] \Rightarrow$
$q \leq 0 \Rightarrow p < q \Rightarrow e[q] < e[p] \vee e[p] = e[q] \wedge n[p] < n[q]$.

Theorem $NormalNotSubNormal$: $\forall p{:}\ float.\ \neg(N[p] \wedge S[p])$.

Theorem $NormalAndSubNormalNotEq$: $\forall p, q{:}\ float.\ N[p] \Rightarrow S[q] \Rightarrow$
$\neg(<R>\ p == q)$.

Theorem $FcanonicUnique$: $\forall p, q{:}\ float.\ C[p] \Rightarrow C[q] \Rightarrow <R>\ p == q \Rightarrow p = q$.

Theorem $FcanonicFormalizeEq$: $\forall p{:}\ float.\ C[p] \Rightarrow norm(p) = p$.

Theorem $FcanonicPosFexpRlt$: $\forall x, y{:}\ float.\ 0 \leq x \Rightarrow 0 \leq y \Rightarrow C[x] \Rightarrow C[y] \Rightarrow$
$e[x] < e[y] \Rightarrow x < y$.

Theorem $FcanonicNegFexpRlt$: $\forall x, y{:}\ float.\ x \leq 0 \Rightarrow y \leq 0 \Rightarrow C[x] \Rightarrow C[y] \Rightarrow$
$e[x] < e[y] \Rightarrow y < x$.

F File Fprop

Theorem $Sterbenz$: $\forall x, y{:}\ float.\ B[x] \Rightarrow B[y] \Rightarrow$
$1/2 * y \leq x \Rightarrow x \leq 2 * y \Rightarrow B[x - y]$.

Theorem $BminusSameExp$: $\forall x, y{:}\ float.\ B[x] \Rightarrow B[y] \Rightarrow$
$0 \leq x \Rightarrow 0 \leq y \Rightarrow e[x] = e[y] \Rightarrow B[x - y]$.

Theorem $BminusSameExpNeg$: $\forall x, y{:}\ float.\ B[x] \Rightarrow B[y] \Rightarrow$
$x \leq 0 \Rightarrow y \leq 0 \Rightarrow e[x] = e[y] \Rightarrow B[x - y]$.

G File FSucc

Definition $FSucc := \lambda x{:}\ float.$
Cases $n[x] = v[b]?$ **of**
$true \Longrightarrow \{nNormMin, 1 + e[x]\}$
$|\ false \Longrightarrow$ **Cases** $n[x] = -nNormMin?$ **of**
$true \Longrightarrow$ **Cases** $e[x] = -d[b]?$ **of**
$true \Longrightarrow \{1 + n[x], e[x]\}$
$|\ false \Longrightarrow \{-v[b], e[x] - 1\}$
end
$|\ false \Longrightarrow \{1 + n[x], e[x]\}$
end
end

In the following we write $(FSuc\ f)$ as $[f + 1]$.

Theorem $FSuccSimpl1$: $\forall x{:}\ float.\ n[x] = v[b] \Rightarrow [x + 1] = \{nNormMin, 1 + e[x]\}$.

Theorem $FSuccSimpl2$: $\forall x{:}\ float.$
$n[x] = -nNormMin \Rightarrow \neg(e[x] = -d[b]) \Rightarrow [x + 1] = \{-v[b], e[x] - 1\}$.

Theorem $FSuccSimpl3$: $[\{-nNormMin, -d[b]\} + 1] = \{1 + -nNormMin, -d[b]\}$.

Theorem $FSuccSimpl4$: $\forall x{:}\ float.$
$\neg(n[x] = v[b]) \Rightarrow \neg(n[x] = -nNormMin) \Rightarrow [x + 1] = \{1 + n[x], e[x]\}$.

Theorem $FSuccDiff1$: $\forall x$: $float$.
$\neg(n[x] = -nNormMin) \Rightarrow <R>\ [x+1] - x == \{1, e[x]\}$.

Theorem $FSuccDiff2$: $\forall x$: $float$.
$n[x] = -nNormMin \Rightarrow e[x] = -d[b] \Rightarrow <R>\ [x+1] - x == \{1, e[x]\}$.

Theorem $FSuccDiff3$: $\forall x$: $float$.
$n[x] = -nNormMin \Rightarrow \neg(e[x] = -d[b]) \Rightarrow <R>\ [x+1] - x == \{1, e[x] - 1\}$.

Theorem $FSuccNormPos$: $\forall a$: $float$. $0 \leq a \Rightarrow N[a] \Rightarrow N[[a+1]]$.

Theorem $FSuccSubnormNotNearNormMin$: $\forall a$: $float$. $S[a] \Rightarrow$
$\neg(n[a] = nNormMin - 1) \Rightarrow S[[a+1]]$.

Theorem $FSuccSubnormNearNormMin$: $\forall a$: $float$. $S[a] \Rightarrow$
$n[a] = nNormMin - 1 \Rightarrow N[[a+1]]$.

Theorem $FBoundedSuc$: $\forall f$: $float$. $B[f] \Rightarrow B[[f+1]]$.

Theorem $FSuccSubnormal$: $\forall a$: $float$. $S[a] \Rightarrow C[[a+1]]$.

Theorem $FSuccPosNotMax$: $\forall a$: $float$. $0 \leq a \Rightarrow C[a] \Rightarrow C[[a+1]]$.

Theorem $FSuccNormNegNotNormMin$: $\forall a$: $float$.
$a \leq 0 \Rightarrow N[a] \Rightarrow \neg(a = \{-(nNormMin), -d[b]\}) \Rightarrow N[[a+1]]$.

Theorem $FSuccNormNegNormMin$: $S[[\{-(nNormMin), -d[b]\} + 1]]$.

Theorem $FSuccNegCanonic$: $\forall a$: $float$. $a \leq 0 \Rightarrow C[a] \Rightarrow C[[a+1]]$.

Theorem $FSuccCanonic$: $\forall a$: $float$. $C[a] \Rightarrow C[[a+1]]$.

Theorem $FSuccLt$: $\forall a$: $float$. $a < [a+1]$.

Theorem $R0RltRleSucc$: $\forall x$: $float$. $x < 0 \Rightarrow [x+1] \leq 0$.

Theorem $FSuccProp$: $\forall x, y$: $float$. $C[x] \Rightarrow C[y] \Rightarrow x < y \Rightarrow [x+1] \leq y$.

Theorem Zeq_Zs: $\forall p, q$: Z. $p \leq q \Rightarrow q < 1 + p \Rightarrow p = q$.

Theorem $FSuccZleEq$: $\forall p, q$: $float$.
$p \leq q \Rightarrow q < [p+1] \Rightarrow e[p] \leq e[q] \Rightarrow <R>\ p == q$.

Definition $FNSucc := \lambda x$: $float$. $[norm(x) + 1]$.

In the following we write $(FNSuc\ f)$ as $[f] + 1$.

Theorem $FNSuccCanonic$: $\forall a$: $float$. $B[a] \Rightarrow C[[a]+1]$.

Theorem $FNSuccLt$: $\forall a$: $float$. $a < [a] + 1$.

Theorem $FNSuccProp$: $\forall x, y$: $float$. $B[x] \Rightarrow B[y] \Rightarrow x < y \Rightarrow [x] + 1 \leq y$.

Theorem $FNSuccEq$: $\forall p, q$: $float$. $B[p] \Rightarrow B[q] \Rightarrow <R>\ p == q \Rightarrow [p] + 1 = [q] + 1$.

Theorem $nNormMimLtvNum$: $(nNormMin) < v[b]$.

Theorem $FSucFSucMid$: $\forall p$: $float$. $\neg(n[[p+1]] = nNormMin) \Rightarrow$
$\neg(n[[p+1]] = -nNormMin) \Rightarrow <R>\ [[p+1]+1] - [p+1] == [p+1] - p$.

Theorem $FNSuccFNSuccMid$: $\forall p$: $float$. $B[p] \Rightarrow \neg(n[[p]+1] = nNormMin) \Rightarrow$
$\neg(n[[p]+1] = -nNormMin) \Rightarrow <R>\ [[p]+1] + 1 - [p] + 1 == [p] + 1 - p$.

H File FPred

Definition $FPred :=$ λx: $float$.
Cases $n[x] = -v[b]$? **of**
$true \Longrightarrow \{-(nNormMin), 1 + e[x]\}$
$|\ false \Longrightarrow$ **Cases** $n[x] = nNormMin$? **of**
$true \Longrightarrow$ **Cases** $e[x] = -d[b]$? **of**
$true \Longrightarrow \{n[x] - 1, e[x]\}$
$|\ false \Longrightarrow \{v[b], e[x] - 1\}$

end
| $false \Longrightarrow \{n[x]-1, e[x]\}$
end
end

In the following we write $(FPred\ f)$ as $[f-1]$.

Theorem $FPredSimpl1$: $\forall x$: $float$.
$n[x] = -v[b] \Rightarrow [x-1] = \{-nNormMin, 1+e[x]\}$.

Theorem $FPredSimpl2$: $\forall x$: $float$.
$n[x] = nNormMin \Rightarrow \neg(e[x] = -d[b]) \Rightarrow [x-1] = \{v[b], e[x]-1\}$.

Theorem $FPredSimpl3$: $[\{nNormMin, -d[b]\}-1] = \{nNormMin-1, -d[b]\}$.

Theorem $FPredSimpl4$: $\forall x$: $float$.
$\neg(n[x] = -v[b]) \Rightarrow \neg(n[x] = nNormMin) \Rightarrow [x-1] = \{n[x]-1, e[x]\}$.

Theorem $FPredFopFSucc$: $\forall x$: $float$. $[x-1] = -[-x+1]$.

Theorem $FPredDiff1$: $\forall x$: $float$.
$\neg(n[x] = nNormMin) \Rightarrow <R>\ x - [x-1] == \{1, e[x]\}$.

Theorem $FPredDiff2$: $\forall x$: $float$.
$n[x] = nNormMin \Rightarrow e[x] = -d[b] \Rightarrow <R>\ x - [x-1] == \{1, e[x]\}$.

Theorem $FPredDiff3$: $\forall x$: $float$.
$n[x] = nNormMin \Rightarrow \neg(e[x] = -d[b]) \Rightarrow <R>\ x - [x-1] == \{1, e[x]-1\}$.

Theorem $FBoundedPred$: $\forall f$: $float$. $B[f] \Rightarrow B[[f-1]]$.

Theorem $FPredCanonic$: $\forall a$: $float$. $C[a] \Rightarrow C[[a-1]]$.

Theorem $FPredLt$: $\forall a$: $float$. $[a-1] < a$.

Theorem $R0RltRleSucc$: $\forall x$: $float$. $0 < x \Rightarrow 0 \leq [x-1]$.

Theorem $FPredProp$: $\forall x, y$: $float$. $C[x] \Rightarrow C[y] \Rightarrow x < y \Rightarrow x \leq [y-1]$.

Theorem $FPredZleEq$: $\forall p, q$: $float$.
$[p-1] < q \Rightarrow q \leq p \Rightarrow e[p] \leq e[q] \Rightarrow <R>\ p == q$.

Definition $FNPred := \lambda x$: $float$. $[norm(x)-1]$.

In the following we write $(FNPred\ f)$ as $[f]-1$.

Theorem $FNPredFopFNSucc$: $\forall x$: $float$. $[x]-1 = -([-x]+1)$.

Theorem $FNPredCanonic$: $\forall a$: $float$. $B[a] \Rightarrow C[[a]-1]$.

Theorem $FNPredLt$: $\forall a$: $float$. $[a]-1 < a$.

Theorem $FNPredProp$: $\forall x, y$: $float$. $B[x] \Rightarrow B[y] \Rightarrow x < y \Rightarrow x \leq [y]-1$.

Theorem $FPredSuc$: $\forall x$: $float$. $C[x] \Rightarrow [[x+1]-1] = x$.

Theorem $FSucPred$: $\forall x$: $float$. $C[x] \Rightarrow [[x-1]+1] = x$.

Theorem $FNPredSuc$: $\forall x$: $float$. $B[x] \Rightarrow <R>\ [[x]+1]-1 == x$.

Theorem $FNPredSucEq$: $\forall x$: $float$. $C[x] \Rightarrow [[x]+1]-1 = x$.

Theorem $FNSucPred$: $\forall x$: $float$. $B[x] \Rightarrow <R>\ [[x]-1]+1 == x$.

Theorem $FNSucPredEq$: $\forall x$: $float$. $C[x] \Rightarrow [[x]-1]+1 = x$.

I File Fmin

Definition $ProjectorP := \lambda P$: $R \to float \to Prop$. $\forall p, q$: $float$. $B[p] \Rightarrow$
$q = P(p) \Rightarrow <R>\ p == q$.

Definition $MonotoneP := \lambda P$: $R \to float \to Prop$. $\forall p, q$: R. $\forall p', q'$: $float$.
$p < q \Rightarrow p' = P(p) \Rightarrow q' = P(q) \Rightarrow p' \leq q'$.

Definition $isMin := \lambda r\colon R.\, \lambda min\colon float.$
$B[min] \wedge min \leq r \wedge \forall f\colon float.\, B[f] \Rightarrow f \leq r \Rightarrow f \leq min.$

In the following we write $(isMin\ r\ f)$ as $f = isMin(r)$.

Theorem $ProjectMin\colon (ProjectorP\ isMin)$.

Theorem $MonotoneMin\colon (MonotoneP\ isMin)$.

Definition $isMax := \lambda r\colon R.\, \lambda max\colon float.$
$B[max] \wedge r \leq max \wedge \forall f\colon float.\, B[f] \Rightarrow r \leq f \Rightarrow max \leq f.$

In the following we write $(isMax\ r\ f)$ as $f = isMax(r)$.

Theorem $ProjectMax\colon (ProjectorP\ isMax)$.

Theorem $MonotoneMax\colon (MonotoneP\ isMax)$.

Theorem $MinEq\colon \forall p, q\colon float.\, \forall r\colon R.$
$p = isMin(r) \Rightarrow q = isMin(r) \Rightarrow <R>\ p == q.$

Theorem $MaxEq\colon \forall p, q\colon float.\, \forall r\colon R.$
$p = isMax(r) \Rightarrow q = isMax(r) \Rightarrow <R>\ p == q.$

Theorem $MinOppMax\colon \forall p\colon float.\, \forall r\colon R.\, p = isMin(r) \Rightarrow -p = isMax(-r).$

Theorem $MaxOppMin\colon \forall p\colon float.\, \forall r\colon R.\, p = isMax(r) \Rightarrow -p = isMin(-r).$

Theorem $MinMax\colon \forall p\colon float.\, \forall r\colon R.$
$p = isMin(r) \Rightarrow \neg(<R>\ r == p) \Rightarrow [p] + 1 = isMax(r).$

Theorem $MinEx\colon \forall r\colon R.\, \exists min\colon float.\, min = isMin(r).$

Theorem $MaxEx\colon \forall r\colon R.\, \exists max\colon float.\, max = isMax(r).$

Theorem $MinBinade\colon \forall r\colon R.\, \forall p\colon float.\, B[p] \Rightarrow$
$p \leq r \Rightarrow r < [p] + 1 \Rightarrow p = isMin(r).$

Theorem $FminRep\colon \forall p, q\colon float.\, q = isMin(p) \Rightarrow \exists m\colon Z.\, <R>\ q == \{m, e[p]\}.$

Theorem $MaxBinade\colon \forall r\colon R.\, \forall p\colon float.\, B[p] \Rightarrow$
$r \leq p \Rightarrow [p] - 1 < r \Rightarrow p = isMax(r).$

Theorem $MaxMin\colon \forall p\colon float.\, \forall r\colon R.$
$p = isMax(r) \Rightarrow \neg(<R>\ r == p) \Rightarrow [p] - 1 = isMin(r).$

Theorem $FmaxRep\colon \forall p, q\colon float.\, q = isMax(p) \Rightarrow \exists m\colon Z.\, <R>\ q == \{m, e[p]\}.$

J File Fodd

Definition $Feven := \lambda p\colon float.\, (Even\ |n[p]|).$

Definition $Fodd := \lambda p\colon float.\, (0dd\ |n[p]|).$

Definition $Feven0 := \forall p\colon float.\, p = 0 \Rightarrow (Feven\ p).$

Theorem $Feven0rFodd\colon \forall p\colon float.\, (Feven\ p) \vee (Fodd\ p).$

Theorem $FoddSuc\colon \forall p\colon float.\, (Fodd\ p) \Rightarrow (Feven\ [p + 1]).$

Theorem $FevenSuc\colon \forall p\colon float.\, (Feven\ p) \Rightarrow (Fodd\ [p + 1]).$

Theorem $FevenFop\colon \forall p\colon float.\, (Feven\ p) \Rightarrow (Feven\ -p).$

Theorem $FoddFop\colon \forall p\colon float.\, (Fodd\ p) \Rightarrow (Fodd\ -p).$

Theorem $FevenPred\colon \forall p\colon float.\, (Fodd\ p) \Rightarrow (Feven\ [p - 1]).$

Theorem $FoddPred\colon \forall p\colon float.\, (Feven\ p) \Rightarrow (Fodd\ [p - 1]).$

Definition $FNodd := \lambda p\colon float.\, (Fodd\ norm(p)).$

Definition $FNeven := \lambda p\colon float.\, (Feven\ norm(p)).$

Theorem $FNoddEq\colon \forall f1, f2\colon float.\, B[f1] \Rightarrow B[f2] \Rightarrow$

$<R>\ f1 == f2 \Rightarrow (FNodd\ f1) \Rightarrow (FNodd\ f2)$.

Theorem $FNevenEq$: $\forall f1, f2$: $float.\ B[f1] \Rightarrow B[f2] \Rightarrow$
$<R>\ f1 == f2 \Rightarrow (FNeven\ f1) \Rightarrow (FNeven\ f2)$.

Theorem $FNevenFop$: $\forall p$: $float.\ (FNeven\ p) \Rightarrow (FNeven\ -p)$.

Theorem $FNoddFop$: $\forall p$: $float.\ (FNodd\ p) \Rightarrow (FNodd\ -p)$.

Theorem $FNoddSuc$: $\forall p$: $float.\ B[p] \Rightarrow (FNodd\ p) \Rightarrow (FNeven\ [p]+1)$.

Theorem $FNevenSuc$: $\forall p$: $float.\ B[p] \Rightarrow (FNeven\ p) \Rightarrow (FNodd\ [p]+1)$.

Theorem $FNevenPred$: $\forall p$: $float.\ B[p] \Rightarrow (FNodd\ p) \Rightarrow (FNeven\ [p]-1)$.

Theorem $FNoddPred$: $\forall p$: $float.\ B[p] \Rightarrow (FNeven\ p) \Rightarrow (FNodd\ [p]-1)$.

K File Fround

Definition $TotalP := \lambda P$: $R \to float \to Prop.\ \forall r$: $R.\ \exists p$: $float.\ p = P(r)$.

Definition $UniqueP := \lambda P$: $R \to float \to Prop.\ \forall r$: $R.\ \forall p, q$: $float.$
$p = P(r) \Rightarrow q = P(r) \Rightarrow <R>\ p == q$.

Definition $CompatibleP := \lambda P$: $R \Rightarrow float \Rightarrow Prop.\ \forall r1, r2$: $R.\ \forall p, q$: $float.$
$p = P(r1) \Rightarrow r1 == r2 \Rightarrow <R>\ p == q \Rightarrow B[q] \Rightarrow q = P(r2)$.

Definition $MinOrMaxP := \lambda P$: $R \to float \to Prop.\ \forall r$: $R.\ \forall p$: $float.$
$p = P(r) \Rightarrow p = isMin(r) \wedge p = isMax(r)$.

Definition $RoundedModeP := \lambda P$: $R \to float \to Prop.$
$(TotalP\ P) \wedge (CompatibleP\ P) \wedge (MinOrMaxP\ P) \wedge (Monotone\ P)$.
In the following we write $(RoundedModeP\ P)$ as $R[P]$ and if P is a rounding mode $(P\ r\ f)$ as $f = P(r)$

Theorem $RoundedProjector$: $\forall P$:?. $R[P] \Rightarrow (ProjectorP\ P)$.

Theorem $MinCompatible$: $(CompatibleP\ isMin)$.

Theorem $MinRoundedModeP$: $R[isMin]$.

Theorem $MaxCompatible$: $(CompatibleP\ isMax)$.

Theorem $MaxRoundedModeP$: $R[isMax]$.

Definition $ToZeroP := \lambda r$: $R.\ \lambda p$: $float.$
$0 \le r \wedge p = isMin(r) \wedge r \le 0 \wedge p = isMax(r)$.

Theorem $ToZeroTotal$: $(TotalP\ \ ToZeroP)$.

Theorem $ToZeroCompatible$: $(CompatibleP\ \ ToZeroP)$.

Theorem $ToZeroMinOrMax$: $(MinOrMaxP\ ToZeroP)$.

Theorem $ToZeroMonotone$: $(Monotone\ \ ToZeroP)$.

Theorem $ToZeroRoundedModeP$: $R[ToZeroP]$.

Definition $ToInfinityP := \lambda r$: $R.\ \lambda p$: $float.$
$r \le 0 \wedge p = isMin(r) \wedge 0 \le r \wedge p = isMax(r)$.

Theorem $ToInfinityTotal$: $(TotalP\ \ ToInfinityP)$.

Theorem $ToInfinityCompatible$: $(CompatibleP\ \ ToInfinityP)$.

Theorem $ToInfinityMinOrMax$: $(MinOrMaxP\ \ ToInfinityP)$.

Theorem $ToInfinityMonotone$: $(Monotone\ \ ToInfinityP)$.

Theorem $ToInfinityRoundedModeP$: $R[ToInfinityP]$.

Theorem $MinUniqueP$: $(UniqueP\ isMin)$.

Theorem $MaxUniqueP$: $(UniqueP\ isMax)$.

Theorem $ToZeroUniqueP$: $(UniqueP\ \ ToZeroP)$.

Theorem $ToInfinityUniqueP$: $(UniqueP\ \ ToInfinityP)$.

Theorem $FnOddNEven$: $\forall n$: $float.\ (FNodd\ n) \Rightarrow \neg((FNeven\ n))$

Theorem $MinOrMaxRep$: $\forall P$:?. $(MinOrMaxP\ \ P) \Rightarrow$
$\forall p, q$: $float.\ q = P(p) \Rightarrow \exists m$: $Z.\ <R>\ q == \{m, e[p]\}$.

Theorem $RoundedModeRep$: $\forall P$:?. $R[P] \Rightarrow \forall p, q$: $float.$
$q = P(p) \Rightarrow \exists m$: $Z.\ <R>\ q == \{m, e[p]\}$.

Definition $SymmetricP := \lambda P$: $R \rightarrow float \rightarrow Prop.\ \forall r$: $R.\ \forall p$: $float.$
$p = P(r) \Rightarrow -p = P(-r)$.

Theorem $ToZeroSymmetric$: $(SymmetricP\ \ ToZeroP)$.

Theorem $ToInfinitySymmetric$: $(SymmetricP\ \ ToInfinityP)$.

L File FroundProp

Definition $Fulp := \lambda p$: $float.\ radix^{e[norm(p)]}$.

In the following we write $(Fulp\ \ f)$ as $Ulp(f)$.

Theorem $FulpComp$: $\forall p, q$: $float.\ B[p] \Rightarrow B[q] \Rightarrow$
$< R > p == q \Rightarrow Ulp(p) == Ulp(q)$.

Theorem $FulpLe$: $\forall p$: $float.\ B[p] \Rightarrow Ulp(p) \leq \{1, e[p]\}$.

Theorem $Fulp\_zero$: $\forall x$: $float.\ x = 0 \Rightarrow Ulp(x) == radix^{-d[b]}$.

Theorem $FulpSucCan$: $\forall p$: $float.\ C[p] \Rightarrow [p + 1] - p \leq Ulp(p)$.

Theorem $FulpSuc$: $\forall p$: $float.\ B[p] \Rightarrow ([p] + 1) - p \leq Ulp(p)$.

Theorem $FulpPredCan$: $\forall p$: $float.\ C[p] \Rightarrow p - [p - 1] \leq Ulp(p)$.

Theorem $FulpPred$: $\forall p$: $float.\ B[p] \Rightarrow p - ([p] - 1) \leq Ulp(p)$.

Theorem $RoundedModeUlp$: $\forall P$:?. $R[P] \Rightarrow \forall p, q$: $float.$
$q = P(p) \Rightarrow |p - q| < Ulp(q)$.

Theorem $FBoundedScale$: $\forall p$: $float.\ \forall n$: $nat.\ B[p] \Rightarrow B[\{n[p], e[p] + n\}]$.

Theorem $FvalScale$: $\forall p$: $float.\ \forall n$: $nat.$
$< R > \{n[p], e[p] + n\} == radix^n * p$.

Theorem $RoundedModeProjectorIdem$: $\forall P$:?. $\forall p$: $float.\ R[P] \Rightarrow$
$B[p] \Rightarrow p = P(p)$.

Theorem $RoundedModeBounded$: $\forall P$:?. $\forall r$: $R.\ \forall q$: $float.\ R[P] \Rightarrow$
$q = P(r) \Rightarrow B[q]$.

Theorem $RoundedModeMult$: $\forall P$:?. $R[P] \Rightarrow \forall r$: $R.\ \forall q, q'$: $float.$
$q = P(r) \Rightarrow B[q'] \Rightarrow r \leq radix * q' \Rightarrow q \leq radix * q'$.

Theorem $RoundedModeMultLess$: $\forall P$:?. $R[P] \Rightarrow \forall r$: $R.\ \forall q, q'$: $float.$
$q = P(r) \Rightarrow B[q'] \Rightarrow radix * q' \leq r \Rightarrow radix * q' \leq q$.

Theorem $RleRoundedR0$: $\forall P$:?. $\forall p$: $float.\ \forall r$: $R.\ R[P] \Rightarrow$
$p = P(r) \Rightarrow 0 \leq r \Rightarrow 0 \leq p$.

Theorem $RleRoundedLessR0$: $\forall P$:?. $\forall p$: $float.\ \forall r$: $R.\ R[P] \Rightarrow$
$p = P(r) \Rightarrow r \leq 0 \Rightarrow p \leq 0$.

Theorem $RoundedModeMultAbs$: $\forall P$:?. $R[P] \Rightarrow \forall r$: $R.\ \forall q, q'$: $float.$
$q = P(r) \Rightarrow B[q'] \Rightarrow |r| \leq radix * q' \Rightarrow |q| \leq radix * q'$.

Theorem $isMinComp$: $\forall r1, r2$: $R.\ \forall min, max$: $float.\ min = isMin(r1) \Rightarrow$
$max = isMax(r1) \Rightarrow min < r2 \Rightarrow r2 < max \Rightarrow min = isMin(r2)$.

Theorem $isMaxComp$: $\forall r1, r2$: $R.\ \forall min, max$: $float.\ min = isMin(r1) \Rightarrow$

$max = isMax(r1) \Rightarrow min < r2 \Rightarrow r2 < max \Rightarrow max = isMax(r2)$.

M File FroundPlus

Theorem $plusExpMin$: $\forall P$:?. $R[P] \Rightarrow \forall p, q, pq$: $float$.
$pq = P(p + q) \Rightarrow \exists s$: $float$. $B[s] \wedge <R>\ s == pq \wedge min(e[p], e[q]) \leq e[s]$.

Theorem $plusExpUpperBound$: $\forall P$:?. $R[P] \Rightarrow \forall p, q, pq$: $float$.
$pq = P(p + q) \Rightarrow B[p] \Rightarrow B[q] \Rightarrow$
$\exists r$: $float$. $B[r] \wedge <R>\ r == pq \wedge e[r] \leq 1 + min(e[p], e[q])$.

Theorem $Fbounded\_Zabs$: $\forall f$: $float$. $B[f] \Rightarrow |n[f]| \leq v[b]$.

Theorem $FboundedShiftLess$: $\forall f$: $float$. $\forall n, m$: nat.
$m \leq n \Rightarrow B[shift(n, f)] \Rightarrow B[shift(m, f)]$.

Theorem $plusExpBound$: $\forall P$:?. $R[P] \Rightarrow \forall p, q, pq$: $float$.
$pq = P(p + q) \Rightarrow B[p] \Rightarrow B[q] \Rightarrow$
$\exists r$: $float$. $B[r] \wedge <R>\ r == pq \wedge$
$min(e[p], e[q]) \leq e[r] \wedge e[r] \leq 1 + min(e[p], e[q])$.

N File FroundMult

Theorem $FboundedOne$: $\forall z$: Z. $-d[b] \leq z \Rightarrow B[\{1, z\}]$.

Theorem $FboundedMbound$: $\forall z$: Z. $\forall m$: nat.
$m \leq radix^{precision} \Rightarrow -d[b] \leq z \Rightarrow \exists c$: $float$. $B[c] \wedge <R>\ c == m * radix^z$.

Theorem $errorBoundedMult$: $\forall P$:?. $\forall p, q, round$: $float$. $R[P] \Rightarrow B[p] \Rightarrow B[q] \Rightarrow$
$-d[b] \leq e[p] + e[q] \Rightarrow round = P(p * q) \Rightarrow$
$\exists r$: $float$. $r == p * q - round \wedge B[r] \wedge e[r] = e[p] + e[q]$.

Theorem $roundedModeLessMult$: $\forall P$:?. $\forall p$: $float$. $\forall r$: R. $R[P] \Rightarrow$
$p = P(r) \Rightarrow \{1, -d[b]\} \leq r \Rightarrow p \leq radix * r$.

Theorem $roundedModeMoreMult$: $\forall P$: $R \Rightarrow float \Rightarrow Prop$. $\forall p$: $float$. $\forall r$: R.
$R[P] \Rightarrow p = P(r) \Rightarrow r \leq \{-1, -d[b]\} \Rightarrow radix * r \leq p$.

Theorem $roundedModeAbsMult$: $\forall P$:?. $\forall p$: $float$. $\forall r$: R. $R[P] \Rightarrow$
$p = P(r) \Rightarrow (\{1, -d[b]\}) \leq |r| \Rightarrow |p| \leq radix * |r|$.

Theorem $multExpMin$: $\forall P$:?. $R[P] \Rightarrow \forall p, q, pq$: $float$. $pq = P(p * q) \Rightarrow$
$\exists s$: $float$. $B[s] \wedge <R>\ s == pq \wedge e[p] + e[q] \leq e[s]$.

Theorem $multExpUpperBound$: $\forall P$:?. $R[P] \Rightarrow \forall p, q, pq$: $float$.
$pq = P(p * q) \Rightarrow B[p] \Rightarrow B[q] \Rightarrow -d[b] \leq e[p] + e[q] \Rightarrow$
$\exists r$: $float$. $B[r] \wedge <R>\ r == pq \wedge e[r] \leq precision + e[p] + e[q]$.

O File Closest

Definition $Closest := \lambda r$: R. λp: $float$. $B[p] \wedge \forall f$: $float$. $B[f] \Rightarrow |p - r| \leq |f - r|$.

Theorem $ClosestTotal$: $(TotalP\ \ Closest)$.

Theorem $ClosestCompatible$: $(CompatibleP\ \ Closest)$.

Theorem $ClosestMin$: $\forall r$: R. $\forall min, max$: $float$. $min = isMin(r) \Rightarrow$
$max = isMax(r) \Rightarrow 2 * r \leq min + max \Rightarrow min = Closest(r)$.

Theorem $ClosestMax$: $\forall r$: R. $\forall min, max$: $float$. $min = isMin(r) \Rightarrow$
$max = isMax(r) \Rightarrow min + max \leq 2 * r \Rightarrow max = Closest(r)$.

Theorem $ClosestMinOrMax$: $(MinOrMaxP\ \ Closest)$.

Theorem $ClosestMinEq$: $\forall r$: R. $\forall min, max, p$: $float$. $min = isMin(r) \Rightarrow$
$max = isMax(r) \Rightarrow 2 * r < min + max \Rightarrow p = Closest(r) \Rightarrow$
$<R>\ p == min$.

Theorem $ClosestMaxEq$: $\forall r$: R. $\forall min, max, p$: $float$. $min = isMin(r) \Rightarrow$
$max = isMax(r) \Rightarrow min + max < 2 * r \Rightarrow p = Closest(r) \Rightarrow$
$<R>\ p == max$.

Theorem $ClosestMonotone$: $(MonotoneP\ \ Closest)$.

Theorem $ClosestRoundedModeP$: $R[Closest]$.

Definition $EvenClosest := \lambda r$: R. λp: $float$.
$p = Closest(r) \wedge FNeven\ p) \vee \forall q$: $float$. $q = Closest(r) \Rightarrow <R>\ q == p$.

Theorem $FNevenOrFNodd$: $\forall p$: $float$. $(FNeven\ p) \vee (FNodd\ p)$.

Theorem $EvenClosestTotal$: $(TotalP\ \ EvenClosest)$.

Theorem $EvenClosestCompatible$: $(CompatibleP\ \ EvenClosest)$.

Theorem $EvenClosestMinOrMax$: $(MinOrMaxP\ \ EvenClosest)$.

Theorem $EvenClosestMonotone$: $(MonotoneP\ \ EvenClosest)$.

Theorem $EvenClosestRoundedModeP$: $R[EvenClosest]$.

Theorem $FnOddNEven$: $\forall n$: $float$. $(FNodd\ \ n) \Rightarrow \neg(FNeven\ \ n)$.

Theorem $EvenClosestUniqueP$: $(UniqueP\ \ EvenClosest)$.

Theorem $ClosestSymmetric$: $(SymmetricP\ \ Closest)$.

Theorem $EvenClosestSymmetric$: $(SymmetricP\ \ EvenClosest)$.

P File ClosestProp

Theorem $ClosestOpp$: $\forall p$: $float$. $\forall r$: R. $p = Closest(r) \Rightarrow -p = Closest(-r)$.

Theorem $ClosestFabs$: $\forall p$: $float$. $\forall r$: R. $p = Closest(r) \Rightarrow |p| = Closest(|r|)$.

Theorem $ClosestUlp$: $\forall p, q$: $float$. $q = Closest(p) \Rightarrow 2 * |p - q| \leq Ulp(q)$.

Theorem $ClosestIdem$: $\forall p, q$: $float$. $B[p] \Rightarrow q = Closest(p) \Rightarrow < R > p == q$.

Theorem $ClosestM1$: $\forall r1, r2$: R. $\forall min, max, p, q$: $float$.
$min = isMin(r1) \Rightarrow max = isMax(r1) \Rightarrow min + max < 2 * r2 \Rightarrow$
$p = Closest(r1) \Rightarrow q = Closest(r2) \Rightarrow p \leq q$.

Q File ClosestPlus

Theorem $errorBoundedPlusLe$: $\forall p, q, pq$: $float$. $B[p] \Rightarrow B[q] \Rightarrow$
$e[p] \leq e[q] \Rightarrow pq = Closest(p + q) \Rightarrow$
$\exists error$: $float$. $<R>\ error == |(p + q) - pq| \wedge B[error] \wedge$
$e[error] = min(e[p], e[q])$.

Theorem $errorBoundedPlusAbs$: $\forall p, q, pq$: $float$. $B[p] \Rightarrow B[q] \Rightarrow$
$pq = Closest(p + q) \Rightarrow$
$\exists error$: $float$. $<R>\ error == |(p + q) - pq| \wedge B[error] \wedge$
$e[error] = min(e[p], e[q])$.

Theorem $errorBoundedPlus$: $\forall p, q, pq$: $float$. $B[p] \Rightarrow B[q] \Rightarrow$
$pq = Closest(p + q) \Rightarrow$
$\exists error$: $float$. $<R>\ error == (p + q) - pq \wedge B[error] \wedge$
$e[error] = min(e[p], e[q])$.

Theorem $plusExact1$: $\forall p, q, r$: $float$. $B[p] \Rightarrow B[q] \Rightarrow$
$r = Closest(p + q) \Rightarrow e[r] \leq min(e[p], e[q]) \Rightarrow <R>\ r == p + q$.

Theorem $plusExact1bis$: $\forall p, q, r$: $float$. $B[p] \Rightarrow B[q] \Rightarrow$
$r = Closest(p + q) \Rightarrow \neg(<R>\ r == p + q) \Rightarrow min(e[p], e[q]) < e[r]$.

Theorem $plusExact2$: $\forall p, q, r$: $float$. $C[p] \Rightarrow B[q] \Rightarrow$
$r = Closest(p + q) \Rightarrow e[r] < e[p] - 1 \Rightarrow <R>\ r == p + q$.

Theorem $plusExactR0$: $\forall p, q, r$: $float.\ B[p] \Rightarrow B[q] \Rightarrow$
$r = Closest(p+q) \Rightarrow <R>\ r == 0 \Rightarrow <R>\ r == p+q.$

Theorem $plusErrorBound1$: $\forall p, q, r$: $float.\ B[p] \Rightarrow B[q] \Rightarrow$
$r = Closest(p+q) \Rightarrow \neg r = 0 \Rightarrow |r-(p+q)| < |r| * 1/2 * radix * 1/v[b].$

R File ClosestMult

Theorem $closestLessMultPos$: $\forall p$: $float.\ \forall r$: $R.$
$p = Closest(r) \Rightarrow 0 \leq r \Rightarrow p \leq 2 * r.$

Theorem $closestLessMultNeg$: $\forall p$: $float.\ \forall r$: $R.$
$p = Closest(r) \Rightarrow r \leq 0 \Rightarrow 2 * r \leq p.$

Theorem $closestLessMultAbs$: $\forall p$: $float.\ \forall r$: $R.$
$p = Closest(r) \Rightarrow |p| \leq 2 * |r|.$

S File Closest2Prop

Local $radix := 2.$

Theorem $FevenNormMin$: $(Even\ nNormMin).$

Theorem $EvenFNSuccFNSuccMid$: $\forall p$: $float.\ B[p] \Rightarrow$
$(FNeven\ p) \Rightarrow <R>\ [[p]+1]+1-([p]+1) == [p]+1-p.$

Theorem $EvenD$: $\forall n$: $nat.\ (Even\ n) \Rightarrow \exists m$: $nat.\ n = 2 * m.$

Theorem $FEvenD$: $\forall p$: $float.\ B[p] \Rightarrow$
$(Feven\ p) \Rightarrow \exists q$: $float.\ B[q] \wedge <R>\ p == 2 * q.$

Theorem $FNEvenD$: $\forall p$: $float.\ B[p] \Rightarrow$
$(FNeven\ p) \Rightarrow \exists q$: $float.\ B[q] \wedge <R>\ p == 2 * q.$

Theorem $AScal2$: $\forall p$: $float.\ <R>\ \{n[p], e[p]+1\} == 2 * p.$

Theorem $div2IsBetween$: $\forall p$: $float.\ \forall min, max$: $float.\ B[p] \Rightarrow$
$min = isMin(1/2 * p) \Rightarrow max = isMax(1/2 * p) \Rightarrow <R>\ p == min + max.$

Theorem $FmultRadixInv$: $\forall x, z$: $float.\ \forall y$: $R.\ B[x] \Rightarrow$
$z = Closest(y) \Rightarrow 1/2 * x < y \Rightarrow 1/2 * x \leq z.$

T File Closest2Plus

Local $radix := 2.$

Theorem $plusUpperBound$: $\forall P$:?$.\ R[P] \Rightarrow \forall p, q, pq$: $float.$
$pq = P(p+q) \Rightarrow B[p] \Rightarrow B[q] \Rightarrow |pq| \leq 2 * min(|p|, |q|).$

Theorem $plusErrorBound2$: $\forall p, q, r$: $float.\ B[p] \Rightarrow B[q] \Rightarrow$
$r = Closest(p+q) \Rightarrow \neg r = 0 \Rightarrow |r-(p+q)| < 2 * 1/v[b] * min(|p|, |q|).$

Theorem Rlt_Rinv2: $\forall r$: $R.\ 0 < r \Rightarrow 1/2 * r < r.$

Theorem $plusClosestLowerBound$: $\forall p, q, pq$: $float.\ B[p] \Rightarrow B[q] \Rightarrow$
$pq = Closest(p+q) \Rightarrow \neg pq == p+q \Rightarrow (1/2 * min(|p|, |q|)) \leq |pq|.$

U File Fast2Sum

Local $radix := 2.$

Variable $Iplus$: $float \rightarrow float \rightarrow float.$

Hypothesis $IplusCorrect$: $\forall p, q$: $float.\ B[p] \Rightarrow B[q] \Rightarrow$
$(Iplus\ p\ q) = Closest(p+q).$

Hypothesis $IplusSym$: $\forall p, q$: $float.\ (Iplus\ p\ q) = (Iplus\ q\ p).$

Hypothesis $IplusOp$: $\forall p, q$: $float$. $-(Iplus\ p\ q) = (Iplus\ -p\ -q)$.

Variable $Iminus$: $float \to float \to float$.

Hypothesis $IminusPlus$: $\forall p, q$: $float$. $(Iminus\ p\ q) = (Iplus\ p\ -q)$.

Theorem $IminusCorrect$: $\forall p, q$: $float$. $B[p] \Rightarrow B[q] \Rightarrow$
$(Iminus\ p\ q) = Closest(p - q)$.

Theorem $ErrorBoundedIplus$: $\forall p, q$: $float$. $B[p] \Rightarrow B[q] \Rightarrow$
$\exists error$: $float$. $<R>\ error == (p + q) - (Iplus\ p\ q) \wedge B[error]$.

Theorem $IplusOr$: $\forall p, q$: $float$. $B[p] \Rightarrow B[q] \Rightarrow$
$<R>\ q == 0 \Rightarrow <R>\ (Iplus\ p\ q) == p$.

Theorem $IminusId$: $\forall p, q$: $float$. $B[p] \Rightarrow B[q] \Rightarrow$
$<R>\ p == q \Rightarrow <R>\ (Iminus\ p\ q) == 0$.

Theorem $IminusOl$: $\forall p, q$: $float$. $B[p] \Rightarrow B[q] \Rightarrow$
$<R>\ p == 0 \Rightarrow <R>\ (Iminus\ p\ q) == -q$.

Theorem $IplusBounded$: $\forall p, q$: $float$. $B[p] \Rightarrow B[q] \Rightarrow B[(Iplus\ p\ q)]$.

Theorem $IminusBounded$: $\forall p, q$: $float$. $B[p] \Rightarrow B[q] \Rightarrow B[(Iminus\ p\ q)]$.

Theorem $IminusInv$: $\forall p, q, r, s$: $float$. $B[p] \Rightarrow B[q] \Rightarrow B[r] \Rightarrow B[s] \Rightarrow$
$<R>\ p == s \Rightarrow <R>\ r == s - q \Rightarrow <R>\ (Iminus\ p\ q) == r$.

Theorem $IminusFminus$: $\forall p, q$: $float$. $B[p] \Rightarrow B[q] \Rightarrow B[p - q] \Rightarrow$
$<R>\ (Iminus\ p\ q) == p - q$.

Theorem $Dekker1$: $\forall p, q$: $float$. $0 \le q \Rightarrow q \le p \Rightarrow B[p] \Rightarrow B[q] \Rightarrow$
$<R>\ (Iminus\ (Iplus\ p\ q)\ p) == (Iplus\ p\ q) - p$.

Theorem $Dekker2$: $\forall p, q$: $float$. $0 \le p \Rightarrow\ -q \le p \Rightarrow p \le (2 * -q) \Rightarrow B[p] \Rightarrow$
$B[q] \Rightarrow <R>\ (Iminus\ (Iplus\ p\ q)\ p) == (Iplus\ p\ q) - p$.

Theorem $Dekker3$: $\forall p, q$: $float$. $q \le 0 \Rightarrow (2 * -q) < p \Rightarrow B[p] \Rightarrow B[q] \Rightarrow$
$<R>\ (Iminus\ (Iplus\ p\ q)\ p) == (Iplus\ p\ q) - p$.

Theorem $MDekker$: $\forall p, q$: $float$. $B[p] \Rightarrow B[q] \Rightarrow |q| \le |p| \Rightarrow$
$<R>\ (Iminus\ (Iplus\ p\ q)\ p) == (Iplus\ p\ q) - p$.

Theorem $Dekker$: $\forall p, q$: $float$. $B[p] \Rightarrow B[q] \Rightarrow |q| \le |p| \Rightarrow$
$<R>\ (Iminus\ q\ (Iminus\ (Iplus\ p\ q)\ p)) == (p + q) - (Iplus\ p\ q)$.

V File EFast2Sum

Local $radix$ **:=** 2.

Variable $Iplus$: $float \to float \to float$.

Hypothesis $IplusCorrect$: $\forall p, q$: $float$. $B[p] \Rightarrow B[q] \Rightarrow$
$(Iplus\ p\ q) = Closest(p + q)$.

Hypothesis $IplusSym$: $\forall p, q$: $float$. $(Iplus\ p\ q) = (Iplus\ q\ p)$.

Hypothesis $IplusOp$: $\forall p, q$: $float$. $-(Iplus\ p\ q) = (Iplus\ -p\ -q)$.

Variable $Iminus$: $float \to float \to float$.

Hypothesis $IminusPlus$: $\forall p, q$: $float$. $(Iminus\ p\ q) = (Iplus\ p\ -q)$.

Theorem $IminusComp$: $\forall p, q, r, s$: $float$. $B[p] \Rightarrow B[q] \Rightarrow B[r] \Rightarrow B[s] \Rightarrow$
$<R>\ p == r \Rightarrow <R>\ q == s \Rightarrow <R>\ (Iminus\ p\ q) == (Iminus\ r\ s)$.

Theorem $EvenBound$: $\forall p$: $float$. $\forall m$: Z. $(Even\ |m|) \Rightarrow$
$((2^{precision}) - 1) \le m \Rightarrow m \le ((2^{1+precision}) - 2) \Rightarrow B[p] \Rightarrow$
$\exists q$: $float$. $B[q] \wedge <R>\ q == \{m, e[p]\}$.

Theorem $LessExpBound$: $\forall p, q$: $float$. $B[p] \Rightarrow B[q] \Rightarrow$

$e[q] \leq e[p] \Rightarrow 0 \leq p \Rightarrow p \leq q \Rightarrow$
$\exists m\colon Z.\ <R>\ \{m, e[q]\} == p \wedge B[\{m, e[q]\}]$.

Theorem $ExtMDekker1\colon \forall p, q\colon float.\ B[p] \Rightarrow B[q] \Rightarrow e[q] \leq e[p] \Rightarrow 0 \leq p \Rightarrow$
$<R>\ (Iminus\ (Iplus\ p\ q)\ p) == (Iplus\ p\ q) - p$.

Theorem $ExtMDekker\colon \forall p, q\colon float.\ B[p] \Rightarrow B[q] \Rightarrow e[q] \leq e[p] \Rightarrow$
$<R>\ (Iminus\ (Iplus\ p\ q)\ p) == (Iplus\ p\ q) - p$.

Theorem $ExtDekker\colon \forall p, q\colon float.\ B[p] \Rightarrow B[q] \Rightarrow e[q] \leq e[p] \Rightarrow$
$<R>\ (Iminus\ q\ (Iminus\ (Iplus\ p\ q)\ p)) == (p + q) - (Iplus\ p\ q)$.

W File ThreeSum2

Local $radix := 2$.

Variable $p, q, r, u, v, w, p', q', r'\colon float$.

Hypothesis $Fp\colon B[p]$.

Hypothesis $Fq\colon B[q]$.

Hypothesis $Fr\colon B[r]$.

Hypothesis $Fu\colon B[u]$.

Hypothesis $Fv\colon B[v]$.

Hypothesis $Fw\colon B[w]$.

Hypothesis $Fp'\colon B[p']$.

Hypothesis $Fq'\colon B[q']$.

Hypothesis $Fr'\colon B[r']$.

Hypothesis $epq\colon e[q] \leq e[p]$.

Hypothesis $eqr\colon e[r] \leq e[q]$.

Hypothesis $uDef\colon u = Closest(q + r)$.

Hypothesis $vDef\colon\ < R > v == (q + r) - u$.

Hypothesis $p'Def\colon p' = Closest(p + u)$.

Hypothesis $wDef\colon\ < R > w == (p + u) - p'$.

Hypothesis $q'Def\colon q' = Closest(w + v)$.

Hypothesis $r'Def\colon\ < R > r' == (w + v) - q'$.

Theorem $bound3Sum\colon \neg(<R>\ r' == 0) \Rightarrow |q' + r'| < (((3 * 2) * 1/v[b]) * |p'|)$.

Theorem $exp3Sum\colon \exists p''\colon float.\ \exists q''\colon float.\ \exists r''\colon float.$
$(B[p''] \wedge B[q''] \wedge B[r''])\ \wedge$
$(<R>\ p'' == p' \wedge <R>\ q'' == q' \wedge <R>\ r'' == r')\ \wedge$
$(e[r] \leq e[r''] \wedge e[r''] \leq e[q''] \wedge e[q''] \leq e[p''])$.

Unit´e de recherche INRIA Lorraine, Technopôle de Nancy-Brabois, Campus scientifique,
615 rue du Jardin Botanique, BP 101, 54600 VILLERS LÈS NANCY
Unit´e de recherche INRIA Rennes, Irisa, Campus universitaire de Beaulieu, 35042 RENNES Cedex
Unit´e de recherche INRIA Rhône-Alpes, 655, avenue de l'Europe, 38330 MONTBONNOT ST MARTIN
Unit´e de recherche INRIA Rocquencourt, Domaine de Voluceau, Rocquencourt, BP 105, 78153 LE CHESNAY Cedex
Unit´e de recherche INRIA Sophia-Antipolis, 2004 route des Lucioles, BP 93, 06902 SOPHIA-ANTIPOLIS Cedex

Éditeur
INRIA, Domaine de Voluceau, Rocquencourt, BP 105, 78153 LE CHESNAY Cedex (France)
http://www.inria.fr
ISSN 0249-6399